

Goldman Sachs

2016 Quant Quest

Patrick Zeng, DoWon Kim, Yash Patel, Jennifer Yin
Princeton University

Presentation Outline

Part 1: Introduction and Background

Part 2: General Approach

Part 3: Technical Approach

Part 4: Results and Discussion

Part 5: Conclusions and Future Work



Part 1: Introduction and Background



Challenge Overview

- **Challenge:** Develop an asymmetrical correlation matrix showing the degree of intercompany relationships between the S&P 500 companies
 - Fully construct 500 by 500 matrix solely using Wikipedia data
 - Use natural language processing techniques on unstructured data to determine magnitude of correlations
 - Value (i, j) is relative link of company j to company i
 - Sum of values across rows, representing relative links of j 's to company i , add up to 1



Economic Intuition: Solution Motivation

- Relationships give perspective on economic linkage between companies
- **Forward direction:** Given substantial change in specific company's financial performance, how significantly are each of the other companies affected?
 - Know which companies a change in Apple's performance would affect most
- **Reverse direction:** How significantly is a specific company affected by changes in each of the other companies?
 - Know which companies to evaluate if we're trying to predict Apple's financial performance



Economic Intuition: The Matrix

- Value (i, j) is **relative link** of company j to company i
 - Measure of how significantly company i is *associated with* company j
- Rows add up to 1: Shows how companies across row are *proportionally linked* to company i
- (i, i) is 0 for all i : To prevent detracting from relationships of other companies to company i

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$



Economic Intuition: The Matrix

- Value (i, j) represents proportional strength of both **forward and reverse direction** economic relationships of company j on company i
 - **Forward:** How a change in i proportionally affects j compared to effect on other companies
 - **Reverse:** How i is affected by a change in j compared to a change in other companies
- Matrix is **asymmetric**: Suppose QCOM (Apple parts manufacturer) is more associated with AAPL than AAPL is associated with QCOM
 - $\text{Value}(\text{QCOM}, \text{AAPL}) > \text{Value}(\text{AAPL}, \text{QCOM})$



Part 2: General Approach




Objectives

- Use data available on Wikipedia pages to generate quantitative basis for calculating link from one company to another
- Quantitatively define a link from one company to another, such that produced results are economically consistent and applicable
- Definition of link should *mimic* and *encapsulate* complex, real-world nature of intercompany relationships



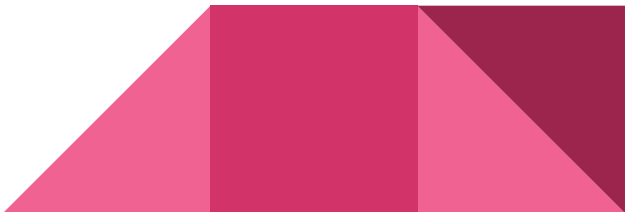
Approach to Implementation

- **Economic factors:** Definition of link should be based on several key economic factors that are easily measurable and comparable between companies, such as market sector, geographic location, and company size
 - **Company history and context:** Definition of link should consider what the company does, as well as any relationships to other companies
 - Wikipedia pages provide both sources of information, through each company's **data box** and **article**
 - **Two-pronged approach:** Use both factor-based and context-based information to calculate links
- 

Part 3: Technical Approach



Concept: Factor-Based Analysis

- **Objective:** Examine and compare key factors of companies we know to be economically significant, such as market sector, location, and company size
 - **Implementation:**
 - Weight key factors individually based on their contributing significance to similarities between companies
 - Use a proprietary **Manual Covariance Analysis (MCA)** technique on values in **data box** to determine measure of correlation based on these factors
 - Produce a matrix that takes into account key factors of companies, forming an economic backbone for next step of analysis
- 

Concept: Content-Based Analysis

- **Objective:** Encapsulate and compare remaining context of companies, including company descriptions and relationships to other companies
- **Implementation:**
 - Scrape **article** content of companies' Wikipedia pages
 - Use **Latent Dirichlet Allocation (LDA)** technique to compare keyword similarities between companies, and establish measures of similarity
 - Produce a final matrix that considers both factor-based and content-based data in determining intercompany relationships
- Partially implemented, project left for future work



Technical Background and Summary

- Technical Analysis implemented with Python
 - Packages: Numpy, Urllib2
- Wikipedia Data Extraction
 - Packages: BeautifulSoup, Pywikibot
- Two Main Approaches:
 - Factor-Based Analysis
 - Content-Based Analysis
- Goal: Output 500 x 500 Correlation Matrix
 - Each (i, j) entry of the matrix indicates relationship of company i with company j
 - Every row should sum to 1 by Normalization



Approach: Factor-Based Analysis

- Analyzed degree of similarity: (AS denotes “asymmetry” and S “symmetry”)
 - **Market Sector:** if two companies are in same sector, correlation will be higher (S)
 - **Sub-Industry:** if same sub-industry, even stronger correlation than solely sector (S)
 - **Geographic Location:** if geographically close, correlation will be stronger (S)
 - **Company Size:** company of larger size will have stronger influence on smaller (AS)
- Data obtained from “List of S&P 500 Companies” Wikipedia Table
- Foundation for content-based analysis
- Factors given different weightings (equation to follow)



Approach: Equation Form

$$\text{Corr}(i,j) = (\text{Sector}(i,j) + \text{SubSector}(i,j)) \cdot \frac{\text{Size}_i}{\text{Size}_j} + e^{-|\text{Region}_i - \text{Region}_j|}$$

$$\text{Sector}(i,j) = \begin{cases} 1 & \text{if } \text{Sector}_i = \text{Sector}_j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{SubSector}(i,j) = \begin{cases} 5 & \text{if } \text{SubSector}_i = \text{SubSector}_j \\ 0 & \text{otherwise} \end{cases}$$



Implementation

- All programming complete in Python, using following notable libraries:
 - **Numpy**: Conveniently manipulating the 2D matrices (optimized speeds)
 - **re**: Regular expressions package in Python -- primarily used for cleaning the data
 - **pywikibot**: Web scraper provided that is capable of more easily parsing Wikipedia pages
 - **urllib2**: Able to obtain the HTML contents of a given page: primarily for LDA
 - **BeautifulSoup**: Used to extract the tags from page for doing concept extraction
- General algorithmic steps of analysis:
 - Extract raw contents of the Wikipedia page (pywikibot)
 - Data cleaning: format the data into recognizable/accurate form (i.e. numbers should be int)
 - Resulted in a 2D array
 - Perform analysis on cleaned data
 - Normalize the matrix



Implementation: Text Cleanup

- A lot of difficulties due to problematic scraping and lengths of rows
- Ended up assuming the most common format of contents and extracting
- Three main cleanup steps:
 - Remove extraneous characters
 - Split the contents of the table
 - Only retain the non-empty cells

```
# 3) Removes the blank array entries
clean_text = [[filter(lambda x : len(x) > 1, [word.encode('utf-8') for word in re.sub('[\n{}\[\]\|]',
    '', row).replace(' ', '\t').split('\t'))])[0] for row in broken_text]
symbols = [company[0].split('Symbol')[-1]
    for company in clean_text]
names = [company[1].split('Symbol')[-1]
    for company in clean_text]

# magic numbers that are used to only extract the wanted values
companies = []
for i in range(0, len(symbols)):
    companies.append([symbols[i]] + clean_text[i][1:])
```

Implementation: Sector and Sub-Industry

- Rationale: external features affect sectors in similar manners
 - Example: Widespread software bug (across all OS) will cause entire software sector decrease
 - Companies linked by their being software
- Similar effect is felt for sub-industry
 - Given more weight than industry as far fewer share same sub-industry
 - Similar to reducing weights of frequent words in NLP algorithms

$$Sector(i, j) = \begin{cases} 1 & \text{if } Sector_i = Sector_j \\ 0 & \text{otherwise} \end{cases}$$

$$SubSector(i, j) = \begin{cases} 5 & \text{if } SubSector_i = SubSector_j \\ 0 & \text{otherwise} \end{cases}$$

Implementation: Sector and Sub-Industry <Code>

- Very simple code
- Determine whether or not the two same sector
- Identical for the sub industry

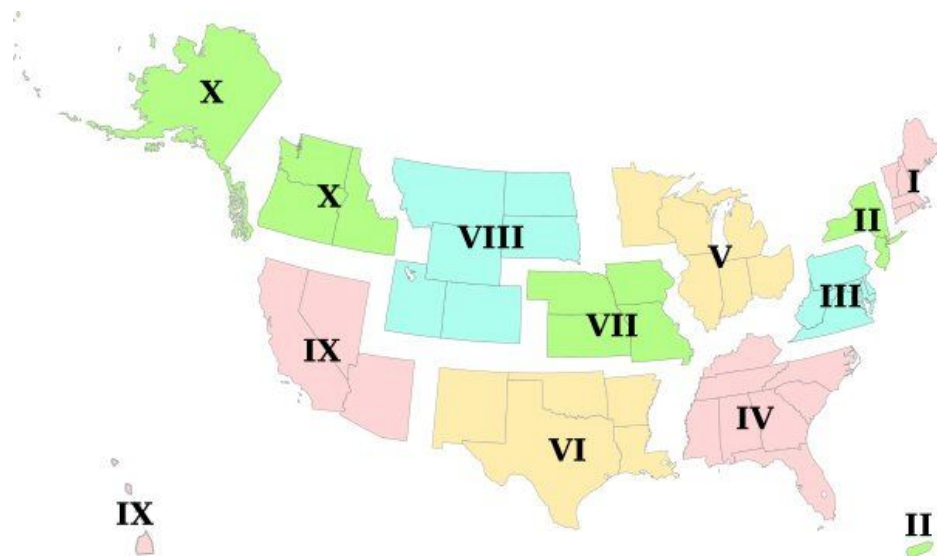
```
# sectors matching adds a SMALL influence
if companyA[SECTOR].strip() == companyB[SECTOR].strip():
    covariance_score += 1

# subsector matching adds a LOT of influence
if companyA[SUB_SECTOR].strip() == companyB[SUB_SECTOR].strip():
    covariance_score += 5
```

Implementation: Geographical Proximity

- Rationale: Geographic location has some fairly uniform features
 - General sentiment towards industries (i.e. strikes in region) cause widespread disdain
- Exact location of the companies not particularly relevant
 - Capturing populace more than influence
 - External factors being shared
- Broke into 10 geographic regions:
 - Standard allotment scheme (right)
 - Regions of greater difference far apart

$$e^{-|Region_i - Region_j|}$$



Implementation: Geographical Proximity <Code>

- Determine the location (from cleaned data)
- Look up region and apply algorithm scaling score

```
def find_region(company):  
    LOCATION = -2  
    text_loc = company[LOCATION].split('<')[0]  
  
    # Eletrical case is only for the dirty data that was extracted  
    # all other cases are non-hard coded  
    if '-' not in text_loc or "Electrical" in text_loc:  
        loc = company[LOCATION].split('<')[0].strip()  
    else:  
        loc = company[LOCATION - 1].split('<')[0].strip()  
  
    if ',' in loc:  
        return get_region(loc.split(',')[1].strip())  
    return get_region(loc)
```

Implementation: Company Size

- Rationale: Size of a company typically indicative of its current influence
 - Asymmetric relation
 - Small company A will have *low* impact on large company B
 - Vice versa for B on A
- Taken as multiplying factor: if no current influence already -- does not change
 - If no match between sectors or location, comparison of size should not increase
- Had multiplicative factor of: $\frac{Size_i}{Size_j}$



Implementation: Company Size <Code>

- Had to extract the Wikipedia page
- Extract the info in “side box”
- Parse through and find employees
 - Default = 25250 (average value)
- With this provide number, calculate

```
def get_employees(company_name):
    site = pywikibot.Site()
    stocks = pywikibot.Page(site, company_name)
    text = stocks.text

    #scrapes information from infobox
    splitter = text.split("{Infobox company\n}")
    if len(splitter) == 1:
        header = splitter[0].encode('utf-8')
    else:
        header = splitter[1].encode('utf-8')

    infobox = header.split("| homepage =")[0].encode('utf-8')
    infobox = re.sub('\n{}\n\[\]', '', infobox)
    table_rows = infobox.split("|")

    # assume default to be 50000
    num_employees = 25250
    for row in table_rows:
        if "employees" in row.lower():
            if len(row.split("=")) == 1:
                num_employees = 25250
                break

            num_employees = row.split("=")[1].strip().split(" ")[0]
            num_employees = re.sub("[^0-9]", "", num_employees)
            if len(num_employees.strip()) <= 1:
                num_employees = 25250
                break

    num_employees = int(num_employees)
    break
```


Implementation: Programming Optimization

- Assuming there are generically n companies ($n = 500$ in this case)
 - Not particularly large dataset -- algorithmic runtime not particularly significant
- Querying online is very slow process
 - Since had to access each of the company's employee size several times, saved these values
 - Values were cached in dictionary: key of company, value of employee size
 - Allowed lookup of values in $O(1)$ time
- Full algorithm runs in $O(n^2)$
 - Have to iterate through each of the companies and compute covariance



Implementation: Content-Based Analysis

- Alternative approach to deduce further relationships between companies
- Utilizes Natural Language Processing techniques, specifically **Latent Dirichlet Allocation**
- Data obtained from each individual company Wikipedia pages
- Intended to add asymmetry to the final correlation matrix
 - Change in Company A \rightarrow Company B \neq Company B \rightarrow Company A
- Analyzes individual word occurrences to retroactively create “topics”
- Assigns probabilistically a topic to each document



Implementation: LDA

- All programming complete in Python, in addition to packages noted earlier
 - **NLTK** - Natural Language Toolkit, for NLP analysis in Python
 - **Stop-Words** - Detects concept-less words in specific languages
 - **Tokenizer / PorterStemmer** - Tools to clean raw text data and remove “stop-words”
- General algorithmic steps of analysis:
 - Extract n paragraphs of text from company-specific Wikipedia pages
 - Data cleaning: Perform analysis on cleaned data
 - *Tokenize* each paragraph into individual words
 - Fits the data via unsupervised modeling onto k set of topics
 - Probabilistically correlates each company-text onto the most-likely topic



Implementation <Code>: LDA

- Takes the text from the Wiki page
 - Extracts the HTML of pages using requests
 - Determines “page text” by value in <p> tags
- Breaks the text into several “tokens”
- Used for performing concept extraction

```
def get_contents(web_page):
    site = pywikibot.Site()
    stocks = pywikibot.Page(site, web_page)
    if len(stocks.text) == 0:
        return ''

    page = requests.get('https:{}'.format(stocks.permlink())).content
    paragraphs = BeautifulSoup(page).findAll('p')
    num_elements = min(2, len(paragraphs))
    page_contents = " ".join([BeautifulSoup(str(par)).p.text
                              for par in paragraphs][0:num_elements]).encode('utf-8')
    return page_contents
```

```
def get_wikitext():
    # Company Wikipedia textfile
    NAME = 1

    companies = get_companies()[1]
    names = [company[NAME].strip() for company in companies]
    names[0] = '3M'

    doc_set = [get_contents(company) for company in names]
    token_set = [tokenize(doc) for doc in doc_set]

    #new_names = names[:5]
    #doc_set = [get_contents(company) for company in new_names]

    return token_set

def tokenize(companytext):
    tokenizer = RegexpTokenizer(r'\w+')
    raw = companytext.lower()
    tokens = tokenizer.tokenize(raw)

    # create English stop words list
    en_stop = get_stop_words('en')
    stopped_tokens = [i for i in tokens if not i in en_stop]

    # Create p_stemmer of class PorterStemmer
    p_stemmer = PorterStemmer()
    return stopped_tokens
```

Implementation <Code>: LDA

- Implements the **LDA model**, specifically *topical analysis*
- Normalizes the probabilities associated with each *topic*
- Generates topic with highest probability for each company

```
def get_topics():  
    token_set = get_wikitext()  
    numpy_token = numpy.array(token_set)  
    # for i in range(0, len(doc_set)):  
    #     for j in range(0, len(doc_set))  
    model = lda.LDA(n_topics = 10, n_iter = 20, random_state = 1)  
    doc_topic = model.numpy_token  
  
    for n in range(15):  
        sum_pr = sum(numpy_token[n,:])  
  
    for n in range(15):  
        topic_most_pr = numpy_token[n].argmax()  
        print("company: {} topic: {}\n{}".format(n, topic_most_pr))  
  
    return topic_most_pr
```

Part 4: Results and Discussion



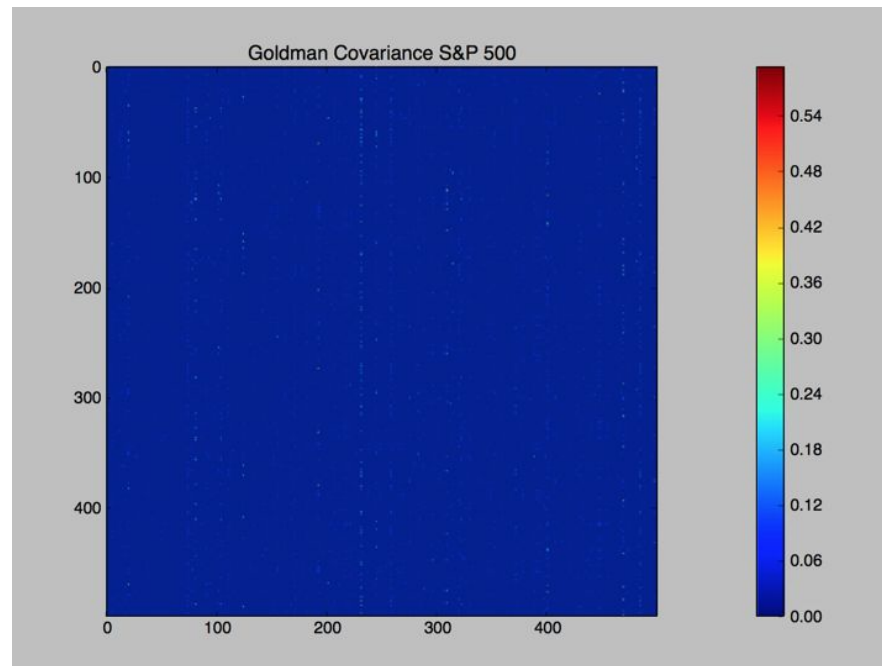
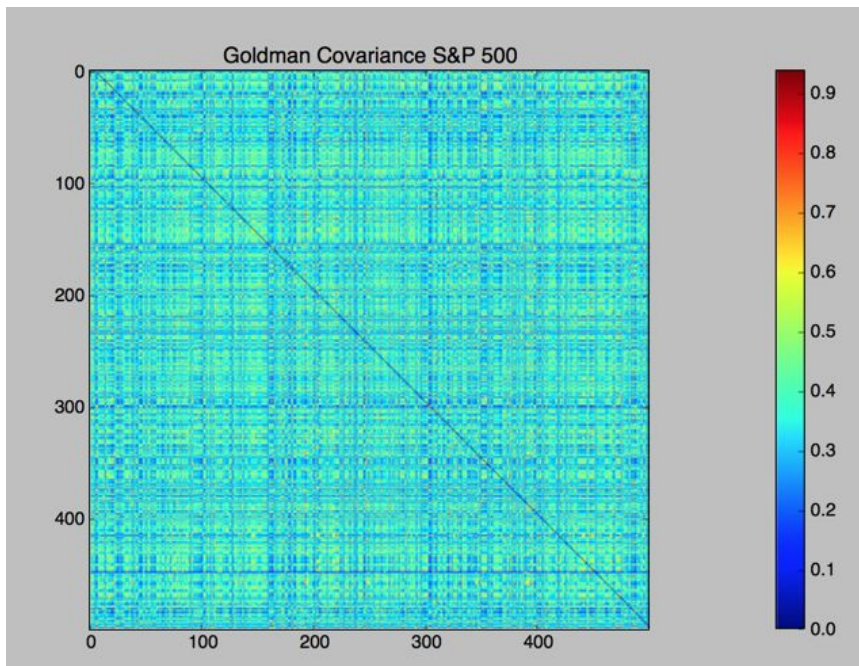
Results: Numerical

- Numerical results: asymmetric
 - Piece of the 500x500 matrix

	A	B	C	D	E	F	G	H	I	J
1	0	0.00164849	0.00164849	1.11E-05	3.02E-05	3.02E-05	0.00555192	0.0002231	0.0002231	3.02E-05
2	0.00130614	0	0.00391842	8.80E-06	2.39E-05	2.39E-05	0.0004805	0.00017677	0.00017677	0.002231
3	0.00364147	0.00364147	0	2.45E-05	6.67E-05	6.67E-05	0.00133962	0.00049282	0.00049282	6.67E-05
4	2.94E-06	2.94E-06	2.94E-06	0	0.04056782	0.01265022	7.99E-06	2.17E-05	2.17E-05	0.00017677
5	0.0001998	0.0001998	0.0001998	1.35E-06	0	0.01090884	7.35E-05	2.70E-05	2.70E-05	3.61E-05
6	0.00012838	0.00012838	0.00012838	8.65E-07	0.02803698	0	4.72E-05	1.74E-05	1.74E-05	2.39E-05
7	0.00203898	0.00203898	0.00203898	0.00010152	3.73E-05	3.73E-05	0	0.00203898	0.00203898	0.0002231
8	0.00038679	0.00038679	0.00038679	0.00014229	7.08E-06	7.08E-06	0.0010514	0	0.002858	0.00038679
9	0.00077381	0.00077381	0.00077381	0.00028467	1.42E-05	1.42E-05	0.00210343	0.00571772	0	0.00077381
10	7.98E-05	7.98E-05	0.00443536	0.00160233	1.46E-06	1.46E-06	0.00021685	0.00058946	0.00058946	7.98E-05
11	0.00063164	0.00063164	0.00063164	3.14E-05	1.16E-05	1.16E-05	0.00171697	0.00063164	0.00063164	8.51E-05
12	0.00015063	0.00015063	0.00015063	0.00302557	2.76E-06	2.76E-06	0.00040947	0.00111304	0.00111304	0.008203
13	0.00012488	0.00012488	0.00012488	8.41E-07	0.00681809	0.00681809	4.59E-05	1.69E-05	1.69E-05	2.29E-05
14	0.00293311	0.00293311	0.00293311	0.00014603	5.37E-05	5.37E-05	0.00797303	0.00293311	0.00293311	0.00038679

Results: Visualization

- Both the same data visualizations -- left is scaled to be more sensitive
- Right exposes lack of symmetry



Results: Natural Language Processing

Partial output of LDA Analysis

```
company: 0    topic: 8  
company: 1    topic: 5  
company: 2    topic: 3  
company: 3    topic: 4  
company: 4    topic: 3
```



Strengths of Approach

- MCA creates factor-based fundamentals for second part of analysis, increasing accuracy and embodying fundamental economic intuition
- LDA analyzes remaining content, capturing diverse, unexpected correlatory factors that MCA may not capture
- Larger companies correlate to each other stronger than smaller companies do; this phenomenon is accounted for by large companies having longer Wikipedia pages. Resulting LDA analysis would thus yield a stronger correlation for larger companies.



Weaknesses of Approach

- Approach can be improved using market sector-specific factors, but we do not have the background to implement such
- Two-part analysis process grants increased accuracy, but at the cost of increased runtime
- Only runs using feature extraction rather than concept



Part 5: Conclusions and Future Work



Future Improvements

- For current algorithm
 - Sector/Sub-industry comparison would extend beyond comparing names directly
 - Specifically: “Technology” sectors might affect “Industrial” sectors
- Determine more accurate metrics for company influence
 - Solely using number of employees seems quite biased
 - Small companies may apply hold influence on niche (Uber in early stages)
- Apply LDA to algorithm
 - Would perform concept extraction on the pages
 - Determine whether any similarity/overlap between the pages



Conclusions

- Found asymmetric covariance matrix for S&P 500 Companies
- Implemented using features extractions with Python and pywikibot scripts
- Learned how to clean raw data available online
 - Many difficulties due to inconsistencies in formatting
- Highly applicable to similar projects due to prevalence of web data



Questions?

Thank you for your attention!

