Project Moksha Final Report

An exploration of Algorithmic Risk Analysis in conjunction with JustProtect

Miguel Amaral, Raquel Julian, Edward Von Leue, Zoe Zumbro

Sponsor: Bryan Urias, JustProtect Information Security Analyst

IDC-4943 Data Analytics Capstone II

Prof. Sravani Vadlamani, PhD.

April 23rd, 2021

Florida Polytechnic University

Table of Contents

Executive Summary 3

Introduction 5

Literature Review 6

Data & Methodology 8

Results 12

Risk Analysis Procedure Document 18

Discussion 19

Future Research 20

Conclusion 22

References 24

Appendix 1 - Risk Analysis Procedure Document 27

Appendix 2 - Python Scripts 43

Appendix 3 - RStudio Scripts 46

Index of Tables

Figure 1. Flowchart of Process 9

Figure 2. Walmart 10K topic model with 4 topics 13

Figure 3. Kroger 10K topic model with 4 topics 14

Figure 4. Target 10K topic model with 4 topics 15

Figure 5. Walmart 10K topic models from 2018-2020 with 6 topics 16

Figure 6. Time-series Walmart risk development from 2011 to 2021 17

Figure 7. Risk assessment comparison between Kroger, Target, and Walmart 19

Figure 8. Future Research in text analysis 20

Executive Summary

JustProtect has a content-agnostic cloud-based platform that centralizes and simplifies assessment processes for companies, and are seeking a process framework that analyzes intel around the supply chain to address possible risks and factors that might affect the supply chain. Team Project Moksha from Florida Polytechnic University has been tasked with creating a process that provides an efficient but detailed overview of risks associated with a company within the supply chain. Risks associated with third-parties can impede progress within the company and disrupt current operations vital in optimizing the business. To prove our process works, we were given the freedom to choose any company to conduct our analysis on.

In this analysis, we took a look at Walmart's reported risks stated within the EDGAR database. EDGAR is an Electronic Data Gathering, Analysis, and Retrieval system, and is the primary system for companies and others submitting documents under the Securities Act of 1933, the Securities Exchange Act of 1934, the Trust Indenture Act of 1939, and the Investment Company Act of 1940. Specifically, financial 10-k document sections titled Item 1a: Risk Factors and Item 7a: Quantitative and Qualitative Disclosures About Market Risk, were used as the primary sources of data to be analyzed. Since the data is in document text format and already has established paragraph breaks, each document analyzed had each row separated into the set paragraph chunks from within both Item 1a and 7a, with column variable names 'Document' and 'Paragraph', completed by computer science interdisciplinaries.

With our target data acquired, the approached methodology is to use text mining by focusing on topic modeling in RStudio using R-packages such as tidytext, tidyverse, topicmodels, dplr, and ggplot. Topic modeling is an unsupervised machine learning that uses Latent Dirichlet Allocation to show the most unique frequent terms being discussed in Walmart's

Form 10-K without assigning predetermined topic names. Visualizations were created to compare the topic's found using standard bar charts and word clouds, which show visual distinction of terms grouped under different topics. Comparison between topic models in Target and Kroger's Form 10-K in relation to Walmart were conducted to see distinctions between the different companies in the same market sector. Additionally, Walmart's Form 10-K's from 2011-2021 were collected and used for the creation of a time series graph that indicates strategic, financial, and operational risks.

The purpose of using topic modeling on Form 10-K's is to see if we can interpret topics found that may indicate that, because this type of company is in this particular sector within the supply chain, they may be prone to particular risks not explicitly stated in the text, but that the accumulation of specific token words actually found in the text can reveal the broader risk that it is categorized under. This is important to help companies steer clear of risks they do not want to expose themselves to, or know themselves that once exposed to the risk, their business will decline. Further research is discussed using supervised machine learning and text classification, as well as potential to generate topic modeling reports for documents in another language. This report will outline progress in the creation of this process, explain more where data was acquired, what is being done with it, and methodologies used to generate results.

Introduction

Project Moksha is intended to create a process designed to present an inclusive risk analysis of companies within various industry segments. It will parse various public documents, including SEC 10-K forms, a comprehensive annual report that covers details on financial performance and risks indications of the company. In order to establish an analysis that facilitates the interpretation of the various risk categories, the project will use Python code language to obtain the most relevant words from the different 10-Ks then use the topic modeling package in RStudio to generate a Word Topic Probability Analysis and a Document Topic Probability Analysis. Using this along with Topic Modeling will provide data that will be presentable in a graphical format and give insight into potential risks that JustProtect's clients may encounter when evaluating other companies for potential business partnerships.

Companies undertake a supplier risk assessment in order to gain relevant information about the companies they intend to partner with because incorporating new vendors into business activities poses the threat of low quality products and services. A risk analysis attempts to find any potential and inherent risks a third-party may possess. These risks may cause financial loss or negatively affect the company's internal operations. Finding and accounting for these risks is important for a company to maintain the integrity of its supply line. Developing a comprehensive supplier risk assessment is critical for any company as it ensures a better understanding of all aspects of risk from potential suppliers.

There are several risk factors that must be considered when conducting a risk assessment of a company. These factors can fall into several categories, such as: strategic, reputational, operational, compliance, and information risk factors. Analyzing such factors can be a complex and laborious process due to the qualitative nature of the data that is usually published and the limited amount of public data available. Therefore, this report is intended to present the progress of Project Moksha in developing a framework that aims to quantify the supply chain disruptions of any company to facilitate that analysis.

Literature Review

The nature of the supply chain is highly risky; there are discrepancies with the information provided by the company such that they do not want particular risks publicly stated for other companies to take advantage of, as well as newly established businesses that are vulnerable to the market. Risk Analysis in general is not strongly structured, with a study conducted in the Oil and Construction industries finding that there was no industry-standard risk analysis process or report (Baker, 1998). Companies that were not involved in large projects were generally unlikely to have formal policies in place to analyze and assess the risks present. This lack of standardization presents a great deal of challenge in comparing and contrasting the risks different companies and industries may be facing. Compounding this sort of problem, is that the nature of a supply chain is prone to a great deal of outside influence and uncertainty, and incorporates both internal and external risks (Ostrom, 2012). Supply chain risk management is particularly prone to generating primarily qualitative risk analysis reports, which are the product of often vague, uncertain, and ill-defined processes (Abdel-Basset, 2019).

Considering how costly the implications of supply chain failures can generate in any given organization, it is crucial to utilize a supply risk assessment methodology that is comprehensive, integrated and accurate (Wu, 2005). However, the qualitative nature of the available data sources restricts the analysis results, as it provides subjective evaluations and generalized metrics. The inclusion of machine learning techniques to categorize risk disclosures is an efficient approach to guarantee economically relevant risks identification (Lopez-Lira, 2019).

For this project, the focus is on companies who've reported risks stated within the EDGAR database. EDGAR is an Electronic Data Gathering, Analysis, and Retrieval system, and is the primary system for companies and others submitting documents under the Securities Act of 1933, the Securities Exchange Act of 1934, the Trust Indenture Act of 1939, and the Investment Company Act of 1940 (Securities and Exchange Commission, 2020). Since our primary source of information is contextual data due to the nature in which the documents are submitted, machine learning and text mining methods were investigated to determine which text mining method will best assist in capturing necessary text data from the EDGAR database.

Kasper Welbers "Text Analysis in R" presents an overview of software libraries, called packages, used to perform text analytics in R, a free, open-source, cross-platform programming environment designed for statistical analysis. Using the *topicmodels* package, we can "look for patterns in the co-occurrence of words and find latent factors (e.g., topics, frames, authors) that explain these patterns—at least mathematically. In terms of inductive reasoning, it can be said that the algorithm creates broad generalizations based on specific observations" (Welbers, 2017). Since EDGAR is composed of documents, we use LDA (Latent Dirichlet Allocation) to fit topic models to these documents by "treating each document as a mixture of topics, and each topic as a mixture of words. This allows documents to "overlap" each other in terms of content, in a way that mirrors typical natural use of natural language.

Data & Methodology

The objective of the project was to analyze companies and determine risks that they may be facing, particularly those that would impact supply chain relations. We decided to approach this by analyzing Items 1A:Risk Factors, and 7A:Quantitative and Qualitative Disclosures about Market Risks, from 10-K reports acquired from the SEC's EDGAR database. The SEC requires companies to publicly disclose reasonable conclusions about anything that could go wrong, such as possible future failures to meet obligations, in order for investors and potential investors to be adequately advised about their investment or future investment in the company. Item 1 describes the business of the company: who and what the company does, what subsidiaries it owns, and what markets it operates in. The risks discussed in Item 1A include recent events, competition, regulations, and labor issues. Other topics in this section may include special operating costs, seasonal factors, or insurance matters. Item 7 discusses the operations of the company in detail and provides an overview of the operational issues that cause increases or decreases in the business. This allows us to group words and phrases into certain categories, which should be related to the kinds of risks a company is facing. Given that 10-K filings have a very similar structure, they are generally easily parsed into a similar format, so that analysis of multiple filings by different companies are capable of being semi-automated. This bulk analysis could allow for a relatively quick comparison of multiple companies to see what risks may set them apart from each other. Done across a multitude of companies in an industry, this could allow seeing what risks a company faces in comparison to its industry. A modified version of this process could potentially be used to do a general industry analysis by using a variety of companies in the industry sector, and doing a similar process with those, to establish the likely risks present throughout that sector, based on common risks and risk topics found with each of the companies.

Figure 1 below lays out the general flow of the risk assessment process we are developing. If the target company has 10-Ks available, we will run them through a JupyterLabs script to find the important words. The list of important words, as well as the surrounding sentence or paragraph that the words are found in will be part of the output. The context around those words will be manually analyzed to make an assessment of the risks that are present. Additionally, the text data will also be run through text mining tools involving topic modeling, which will present a group of words that are estimated by Latent Dirichlet allocation (LDA) to be words that relate to a similar topic. These words will then be used to theorize the topic that is being addressed by those words. Those topics will be used to estimate what risk is being addressed. Lastly, other risks can be estimated using content found on OSI. The results of all these steps will be compiled into a report.

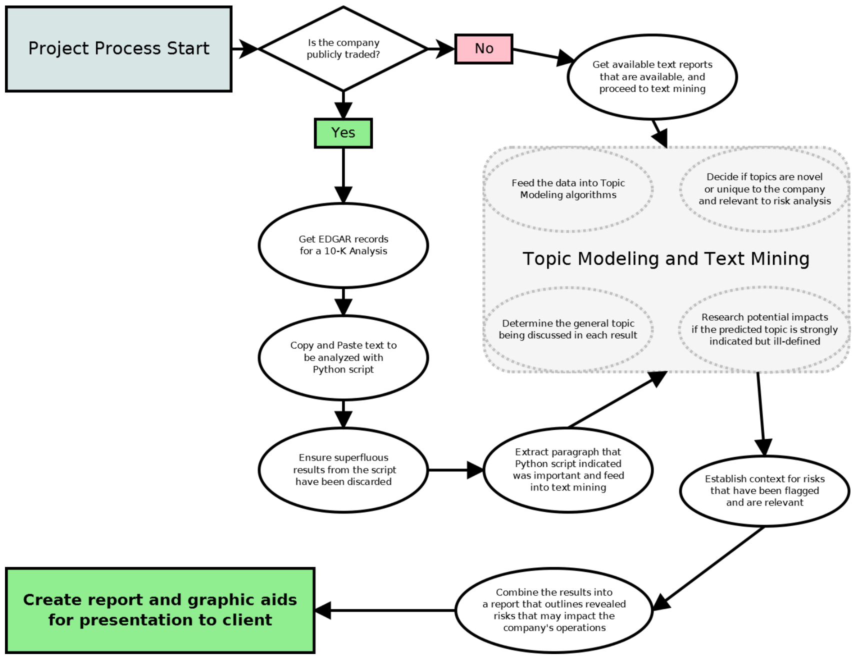

Project Process Start
Is the company publicly traded?
No
Get available text reports that are available, and proceed to text mining
Yes
Get EDGAR records for a 10-K Analysis
Copy and Paste text to be analyzed with Python script
Ensure superfluous results from the script have been discarded
Extract paragraph that Python script indicated was important and feed into text mining
Feed the data into Topic Modeling algorithms
Decide if topics are novel or unique to the company and relevant to risk analysis
Topic Modeling and Text Mining
Determine the general topic being discussed in each result
Research potential impacts if the predicted topic is strongly indicated but ill-defined
Establish context for risks that have been flagged and are relevant
Combine the results into a report that outlines revealed risks that may impact the company's operations
Create report and graphic aids for presentation to client

Figure 1. Flowchart of Process

The scripts referenced below are located in the appendices of the Risk Analysis Process Document located in Appendix 1. The starting point for the process is acquiring the 10-K filings for the identified company. The text of those files are then copied to a text file and fed into a python script that has been written by the Computer Science Interdisciplinary Team. Their script then parses the text and places it into chunks that can be analyzed with the Term Frequency-Inverse Document Frequency technique. This identifies the terms that are more important to a document than the rest. It gives a value that increases proportionally to the number of times a word appears in a document, and offset by the number of documents that the word appears in, which helps adjust for the fact that some words appear more frequently in general. When combined with a list of precompiled stop-words, which are the aforementioned overly common words, we are given a list of words that are determined by the python algorithm to be the most important words in the document. We currently limit this to the top 100 results. These words are then used to guide the next steps of the process.

Item 1A and Item 7A were then pulled from the Python script and imported into RStudio, which was used in implementing the ‘topicmodels’ package, a machine learning and natural language processing model that discovers abstract topics that occur in a collection of documents. This process involves counting words and grouping word patterns to infer topics with unstructured data. We also incorporate LDA (Latent Dirichlet Allocation), a popular topic modeling algorithm that allows a set of observations to be explained by unobserved groups that explain why some parts of the data are more similar than others. For this project, we extract the per-topic-per-word probabilities by turning EDGAR financial documents into a one-topic-per-term-per-row format. This allows the model to “compute the probability of that term being generated from that topic” (Robinson, 2020) which enables us to focus on words in the document that are risk-associated for targeted results. The goal is to map these documents into topics in such a way that each document is mostly captured by the topics generated from it. By using topic modeling and LDA, it can tell us which topics are prevalent in the sections being analyzed (Item 1A and Item 7A) by observing all the words in it, and producing topic distributions.

To establish a measurement of risk comparison between companies in the same industry, it was necessary to identify a common quantitative value per topic. Therefore, the average beta value of the set of words for each topic was used to compare how much time each company spent discussing a specific risk. It was assumed that a company is relatively more exposed to a risk according to how much time it was addressed in the text. The same RStudio script was adapted to extract the average beta per risk factor for Walmart, Target, and Kroger, and generate a grouped bar chart using the ggplot2 Package. Taking into consideration that the RStudio code does not give us labels for the topics and it is unlikely to get an identical set of words, it was necessary to establish a method to associate topics. As the topics are easily interpretable due to the most frequent words, it was assumed that risk factors with multiple similar words are highly likely to refer to the same risk. Once the same risks were grouped, the code creates a visualization that illustrates the intensity of each singular risk compared to its competitors. The result is probabilistic evidence of the current situation of each company and provides to the end user a significant orientation to assess the supply risk assessment between their potential partners or competitors.

The same methodology of extracting the beta average to determine the intensity of the risk was utilized to establish a time-series graph of three types of risk based on Walmart 10Ks reports from 2011 to 2021. In order to create the visualization, an RStudio code was developed to determine the topic modeling over the eleven years, and the Financial, Operation, and Strategic risks were manually grouped to establish the analysis. This type of analysis allows us to interpret the fluctuations of the risk categories over time. Furthermore, it provides a clear understanding of the company's trends and exposes potential disruption in the supply chain.

Results

Semester 1

Our task for this semester was to analyze 3 companies within the same economic sector. Walmart, Target, and Kroger are competing companies in the grocery/retail related industry, which are especially liable to many risks within the third-party companies they have a business partnership with in the supply chain. By independently evaluating Item 1A and Item 7A in each 10K's, we are able to assign topic models to the document. The framework delivered to JustProtect would see this portion completed by an industry expert who is familiar with company financial filings and reporting terminology.

Since we did not have access to an industry expert at the time of this analysis, the methodology in evaluating these topics was to take the top 4-5 words independently within each topic whose beta value was among the highest, and simply search the Web using those words, to find business related risks whose definition had a combination of these words in it or related. We found this could be beneficial in our current state because search engines have their own keyword and search pattern system, and by searching for these terms altogether, will show results of how and within what context they are collectively used in. With a high beta value, topics are made up of the most words in the document, so, the word with the highest beta value within each topic holds more weight in that topic amongst the other words assigned to that group.

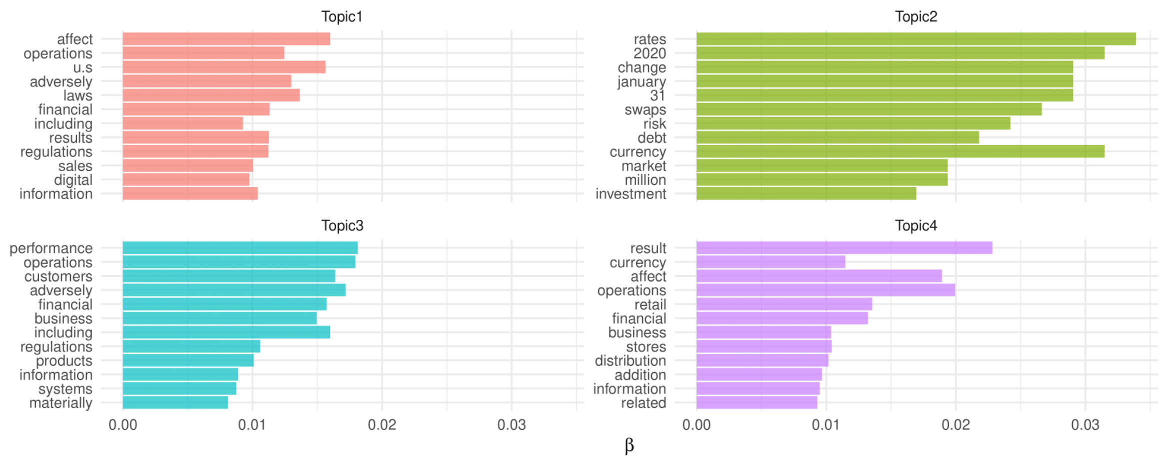


Figure 2. Walmart 10K topic model with 4 topics

In evaluating Walmart’s 10K assigning 4 topics, it can be said that topic 1 is *Regulatory Risk* because of terms such as ‘laws’, ‘operations’, ‘financial’, and ‘regulations’; regulatory risk is defined as “a change in laws and regulations that materially impact a security, business, sector, or market” (Investopedia). Topic 2 indicates *Currency Rate Risk* because of terms such as ‘currency’, ‘change’, ‘swaps’, ‘risk’, and ‘debt’; currency rate risk is defined as “the possibility that an investment's value may decrease due to changes in the relative value of the involved currencies” (Investopedia). Topic 3 shows *Operational Risk* because of terms such as ‘performance’, ‘operations’, ‘financial’ ‘materially’ and ‘systems’; operational risk is defined as “the risk of loss resulting from inadequate or failed internal processes, people, controls, systems or from external events” (Basel Committee on Banking Supervision). Topic 4 is *Regulatory and Distribution Risk* because of terms such as ‘operations’, ‘retail’, ‘stores’, ‘distribution’, ‘financial’ and ‘result’; as regulatory risk is defined above, distribution risk “results in risks to a distribution channel, to the insurer's business, and ultimately to its financial sustainability” (Gutterman).

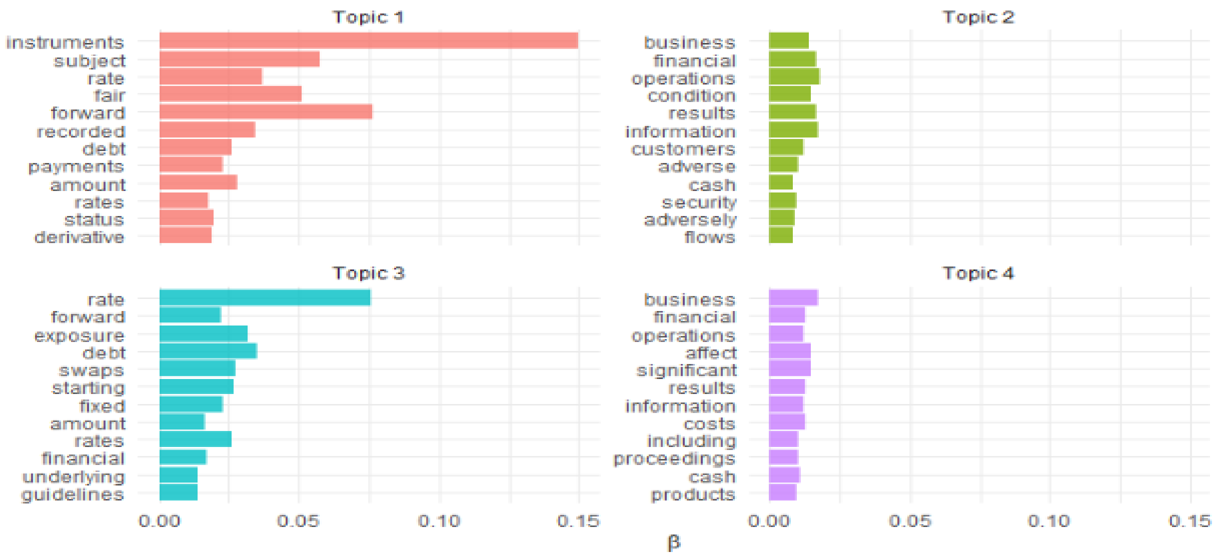


Figure 3. Kroger 10K topic model with 4 topics

In evaluating Kroger's 10K documents and assigning topics, it can be said that Topic 1 is *Contract Agreement Risk* because of terms such as 'instruments', 'subject', 'forward', 'rate' and 'fair'; contract risk is defined as "the chance of facing losses as a result of the buyer not fulfilling the terms of a contract, not including if the buyer is incapable of paying" (Khan). Topic 2 and Topic 4 are similar in nature, visually from the bar graphs as well as in their topic terms. We named these topics *General Business Risk* because of terms like 'business', 'financial', 'condition', 'cash', and 'operations'; business risk is defined as "a chance of incurring losses or less profit than expected" (Toppr). Lastly, Topic 3 is *Interest Rate Risk* because of terms such as 'rate', 'debt', 'fixed', 'exposure', 'starting', and 'underlying'; interest rate risk is defined as "the probability of a decline in the value of an asset resulting from unexpected fluctuations in interest rates" (Corporate Finance Institute).

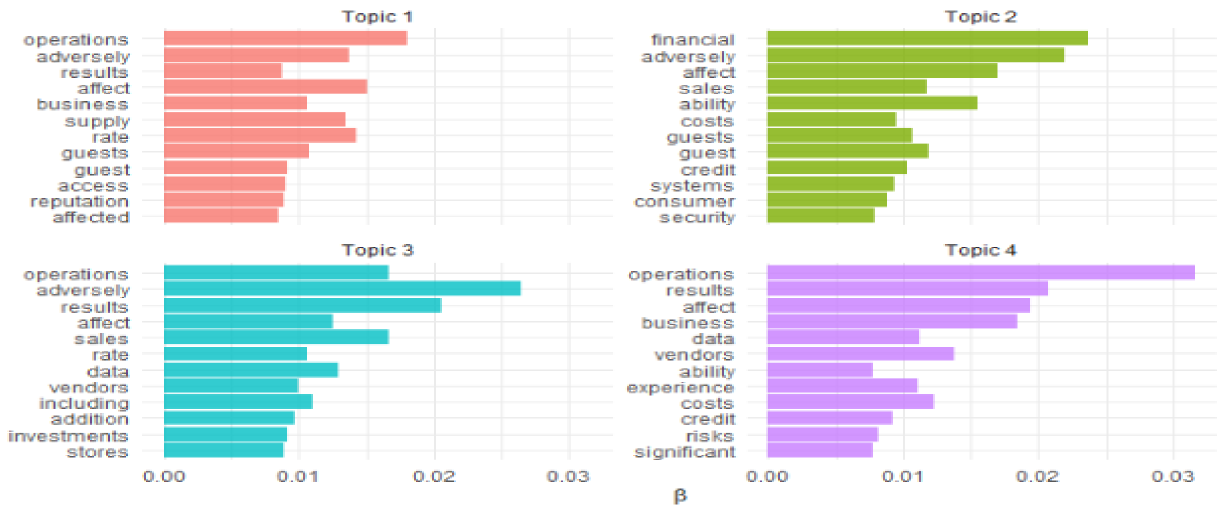


Figure 4. Target 10K topic model with 4 topics

In evaluating Target's 10K documents and assigning topics, Topic 1 is *Operational and Reputational Risk* because of terms such as 'operations', 'supply', 'rate', 'guest', 'affect' and 'reputation'; as operational risk is defined above, reputational risk is defined as "the threat to the profitability or sustainability of a business or other entity that is caused by unfavorable public perception of the organization or its products or services" (Wigmore). Topic 2 is *Financial Risk* because of the terms 'financial', 'sales', 'costs', 'guest', and 'credit'; financial risk is defined as "any of various types of risk associated with financing, including financial transactions that include company loans in risk of default" (Dbpedia). Topic 3 is *Information Risk* because of terms such as 'adversely', 'data', 'results', 'sales', and 'vendors'; information risk is defined as "a calculation based on the likelihood that an unauthorized user will negatively impact the confidentiality, integrity, and availability of data that you collect, transmit, or store" (Fasulo). Topic 4 is *Information and Operational Risk* because of terms such as 'operations', 'business', 'data', 'vendors', and 'results'.

<u>Semester 2</u>

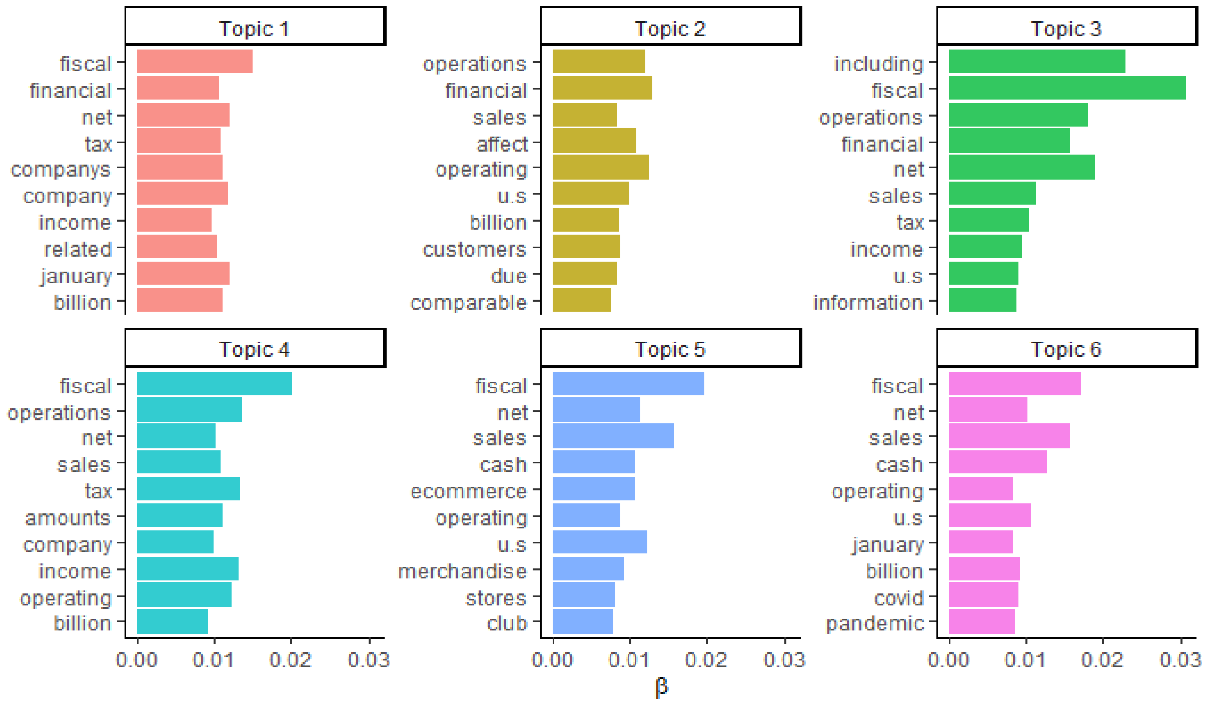


Figure 5. Walmart 10K topic models from 2019-2021 with 6 topics

To incorporate the process from the first semester, we decided to increase the scale of the amount of documents processed. Shown in Figure 5 are topics from Walmart's Form 10K for 2019, 2020, and 2021. After merging these documents and applying LDA, the newly created model was set to find 6 topics. As the number of topics are up for interpretation, the increase from 4 topics to 6 topics is because the addition of more documents evaluated more words, so to apply a topic at least more than 4 topics seemed appropriate. The word *'fiscal'* appears in almost all topics, which is logical given that SEC Form 10-K's discuss elements related to financial matters. Looking at the other terms, we can see a distinct differentiation between Topic 5 and Topic 6; Topic 5 discusses monetary terms in sales and ecommerce operations through the words *'sales' 'cash' 'ecommerce' 'merchandise'* and *'stores'*. Topic 6 discusses monetary terms operating during the COVID-19 pandemic through the words *'sales' 'cash' 'covid'* and *'pandemic'*. Addressing these topics with a background of risk, it can be said that Walmart experiences risk within their sales/cash operations through their brick-and-mortar locations as well as through ecommerce operations on their website. Topic 6 addressing the COVID-19 pandemic presents a disclosure of risk surrounding uncertainties around COVID-19 policies. These are some potential common topics addressed throughout Walmart's reports.

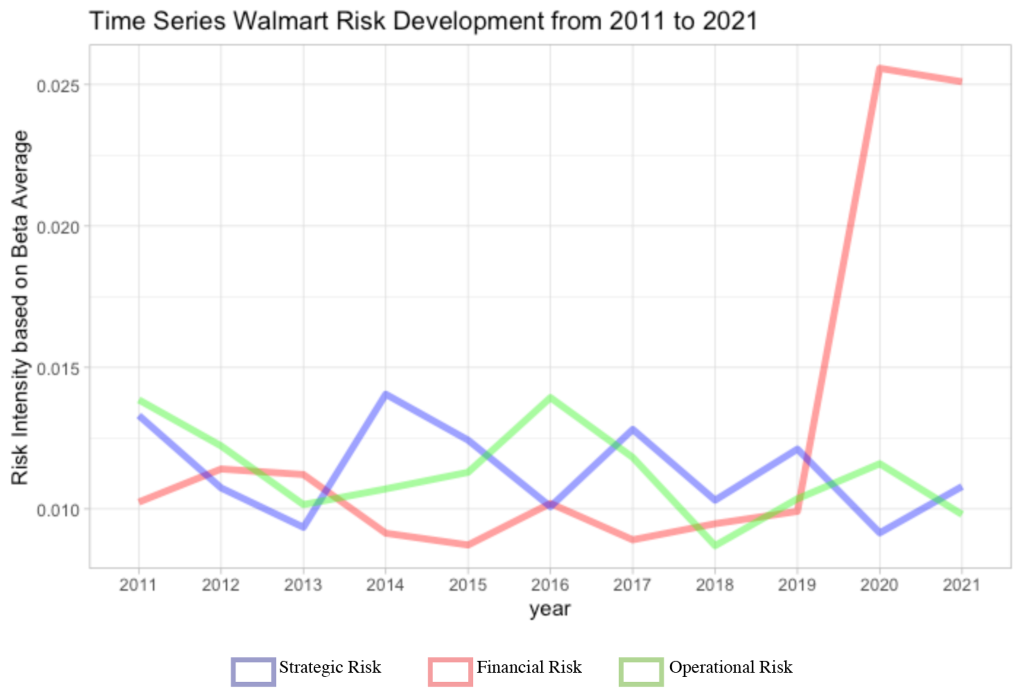


Figure 6. Time-series Walmart risk development from 2011 to 2021

Figure 6 illustrates the main Walmart risk fluctuations over the past eleven years. As we can observe, the operational and strategic risks are relatively constant over time, with periodic ups and downs. On the other hand, in respect of the financial risk, it is possible to affirm that the company spent more time discussing financial aspects and potential financial risks in 2020 and 2021 in the 10K reports. The rapid increase in the beta average could have happened due to many reasons, however, the graph orients the end-user that a sudden change in the financial sector occurred and must be further explored.

One possible reason for this spike is the COVID-19 pandemic mentioned in Walmart's 2020-2021 Form 10K, which may have caused great financial uncertainty surrounding Walmart and their finances, given that most of their profits before the pandemic came from their physical store locations. The switch to focus more on online services and delivery may have caused them to disclose financial uncertainties because of abnormal operations that year. Conversely, as financial risk increases, strategic risk declines, which could indicate further measures of uncertainty, such that, a spike in financial risk caused strategic decisions to decline; perhaps there was a noticeable decline in strategic partnerships with other companies for a period of time in order to ensure their own processes and business operations were performing at its targeted rate to ensure profitability. But again, this must be further explored through other means of documentation.

Risk Analysis Procedure Document

In addition to the above, we have been requested to develop a framework document to guide a risk analysis procedure. This document is attached in Appendix 1. The framework is intended as a supplement to ISO 31010. It is a step-by-step guide for conducting a risk analysis assessment using the technique we've developed. This details the steps to find forms of publicly-available data, how to feed them into a script that will show the document tf-idf, and which is used to aid LDA and Topic Modeling analysis. For the supplied document, these steps will generate a list of important words used to identify additional stop words that hinder the clarity of the analysis, and use the remaining words left in the document to generate a set of graphs using Topic Modeling analysis. These would then be used to create a report on the company. This report could be used for further research and implementation to present a Risk

Analysis assessment to a client of JustProtect. Optionally, this process could be iterated to conduct this analysis for each of a company's suppliers, and/or many different companies within an industry.

Discussion

In order to test the effectiveness of the system and create a comparative analysis between the results, three publicly-traded companies were chosen. Items 1A Risk Factors and 7A Quantitative and Qualitative Risk Disclosures About Market Risk of 10Ks from Microsoft, Visa, and Walmart were extracted and converted to a csv file. According to the established process flow, the same text input was used in the JupyterLabs script to generate the Word Frequency Index and in the RStudio script to generate a Word Topic Probability Analysis and a Document Topic Probability Analysis.

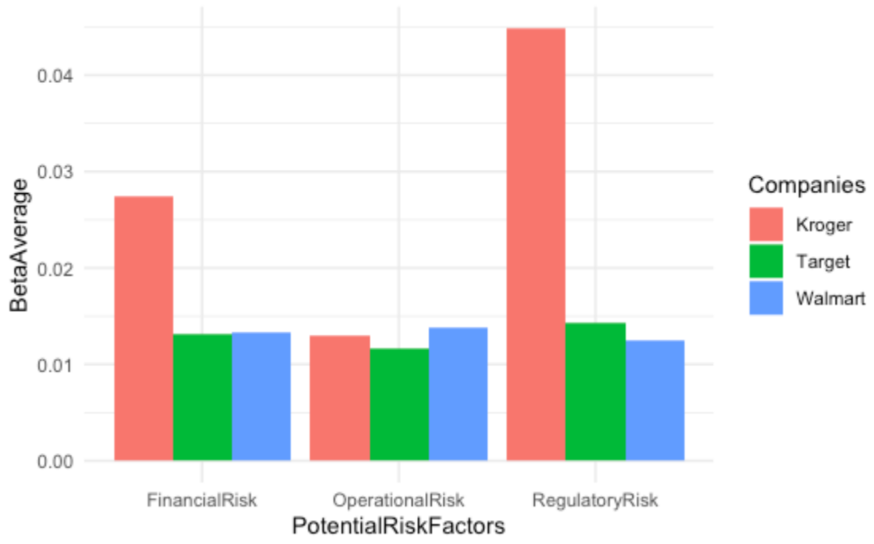
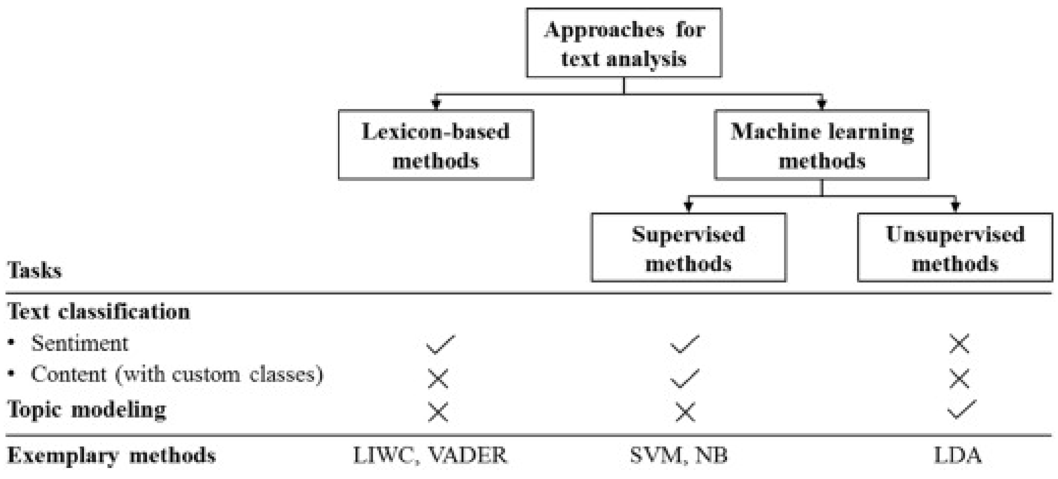


Figure 7. Risk assessment comparison between Kroger, Target, and Walmart

Figure 7 demonstrates a risk assessment comparison between Walmart, Target, and Kroger using the average beta per topic as the quantitative value used to identify the intensity of each category. The risks were manually linked by the similarities in the words set for each topic. For instance, in order to elaborate the Financial Risk category, the values from the Target Financial Risk, the Kroger Interest Rate Risk, and the Walmart Currency Rate Risk were utilized. As we can observe, the graph indicates a higher probability of regulatory disruptions within Kroger's supply chain when compared to its competitors. This analysis is based on the time that the company spent discussing each particular subject. Likewise, it is possible to say that Kroger is more exposed to threats in the financial sector compared to Target and Walmart. On the other hand, considering the non-absolute nature of qualitative analysis and topic modeling, it is reasonable to affirm that all companies are exposed in a similar intensity to Operational Risks.

Future Research



Tasks	Lexicon-based methods	Supervised methods	Unsupervised methods
Text classification			
• Sentiment	✓	✓	✗
• Content (with custom classes)	✗	✓	✗
Topic modeling	✗	✗	✓
Exemplary methods	LIWC, VADER	SVM, NB	LDA

Figure 8. Future Research in text analysis

Since topic modeling is an unsupervised machine learning method, further research in text analysis suggests using text classification, a supervised machine learning method, after topic modeling to assign topic labels to each textual document by using the result of the previous step as target labels. Topic modeling helped capture the essence of elements discussed in SEC Form 10K's, so to advance this project, tagging these topics with a label that the algorithm can train and retain on the more documents are added to it, which can help bring a more structured view of topics discussed.

There are some additional steps that could be integrated into this process to make it more informative for the end user if ever given the chance. For example, research could be done to locate the company's suppliers so that the supplier's financial information can be used to assess their risk assessments as well. The end user can then recognize their lowest and highest risk suppliers to better help coordinate their business relationships. Another possible research outlet would be to use topic modeling to analyze news outlets to see what is being said about the company in the media. This allows the stakeholders to see what kind of risks politicians, customers, and any other end user might be associating with the business. The project could also expand to incorporate financial documents written in other languages by having a piece of code translate the document and then run it through the same process. This could allow JustProtect to expand their customer base as well as observe trends from all around the world. The work that has been completed thus far is only scratching the surface, there are a multitude of outlets that this project can expand to in the future.

Conclusion

In the beginning, Project Moksha was faced with a broad scope that took about a month to narrow down. Since then, the project has come a long way. By analyzing the 10K financial reports from publicly traded companies through a variety of scripts, several pieces of data have been found, like words deemed important by LDA and its context, topic models and inverse document frequency modeling, and general business risks of which have been publicly disclosed (such as data branches). The framework deliverable has been flushed out so that JustProtect can use it as a guideline to help their customers. All of the Topics have been categorized and assigned to risk types from the topic models. This took out any bias caused by assumptions or using intuition. We were also able to create an extensive list of stop words to input into the scripts that would ultimately eliminate any unnecessary or repetitive words found in the topic models. As a final deliverable to JustProtect, we will be writing a report analyzing the results of the risk assessment from the three companies we have done the initial research on. This will go in hand with the process flowchart and this report (that can be utilized by any future team that may add on to this project).

This was a very educational project that has allowed all of us to use our education in a real world example as well as in a very professional setting. Working with JustProtect has been an excellent opportunity for us as they were very helpful in guiding us and helping us meet our goals.

References

Abdel-Basset, M.. (2019). *A framework for risk assessment, management and evaluation: Economic tool for quantifying risks in supply chain*. Future Generation Computer Systems, 90, 2019.

Baker, S.. (1998). *Techniques for the Analysis of Risks in Major Projects*. The Journal of the Operational Research Society, vol. 49, 572.

Basel Committee on Banking Supervision. (2011, June). Principles for the Sound Management of Operational Risk. Retrieved April, 2021, from https://www.bis.org/publ/bcbs195.pdf

CFI Education Inc. (2020, February). Interest Rate Risk - Definition, How to Mitigate the Risk. Retrieved April, 2021, from https://corporatefinanceinstitute.com/resources/knowledge/finance/interest-rate-risk/

DBpedia. (n.d.). About: Financial Risk. Retrieved April, 2021, from https://dbpedia.org/page/Financial_risk

Fasulo, P. (2019, December). What is IT Risk Management? A Complete Guide. Retrieved April, 2021, from https://securityscorecard.com/blog/what-is-information-risk-management

Ganti, A. (2020, November). Foreign Exchange Risk Definition. Retrieved April, 2021, from https://www.investopedia.com/terms/f/foreignexchangerisk.asp

Gutterman, S. (2016). 9 - Distribution Risks. In 1310448073 963483078 International Actuarial Association (Author), IAA Risk Book. Insurance Regulation Committee of the IAA.

Hayes, A. (2021, January). Regulatory Risk Definition. Retrieved April, 2021, from https://www.investopedia.com/terms/r/regulatory_risk.asp

Khan, M. S. (2019, February). Legal Risk in the Context of Risk Management. Retrieved April, 2021, from https://www.linkedin.com/pulse/legal-risk-context-management-mohammed-salman-khan/

Lopez-Lira, A. (January 2019). *Risk factors that matter: textual analysis of risk disclosures for the cross-section of return.* The Wharton School. https://www.loyola.edu/-/media/join-us/baltimore-finance/documents/2019-presentation-files/lopez-lira.ashx?la=en

Nordine, Justin. *OSINT Framework.* Retrieved Oct 12, 2020, from http://osintframework.com

Ostrom, L. T., & al, E. (2012). *Risk Assessment: Tools, Techniques, and Their Applications*. John Wiley & Sons.

pandas development team. (2020). *User Guide - pandas 1.1.4 documentation*. Retrieved Oct 12, 2020, from https://pandas.pydata.org/docs/user_guide/index.html

Robinson, J. S. A. D. (2020, November 10). *6 Topic modeling | Text Mining with R*. Tidy Text Mining. https://www.tidytextmining.com/topicmodeling.html scikit-learn developers. (2020). *User guide: contents - scikit-learn 0.23.2 documentation*. Retrieved Oct 19, 2020, from https://scikit-learn.org/stable/user_guide.html

Securities and Exchange Commission. (2020). *About EDGAR*. SEC.Gov. https://www.sec.gov/edgar/about

Securities Exchange Commission. (2019). *Accessing EDGAR Data*. Retrieved Sept 21, 2020, from https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm

Toppr. (2019, December). Nature of Business Risk: Meaning, Causes and Types of Risks, Examples. Retrieved April, 2021, from https://www.toppr.com/guides/business-studies/nature-and-purpose-of-business/nature-of-business-risk/

Verizon. (2020). *Verizon 2020 Data Breach Investigations Report.* Retrieved Oct 5, 2020, from https://enterprise.verizon.com/resources/reports/dbir/2020/cheat-sheet/

Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). *Text Analysis in R*. Taylor & Francis. https://kenbenoit.net/pdfs/text_analysis_in_R.pdf

Wigmore, I. (2017, May). What is reputation risk? - Definition from WhatIs.com. Retrieved April, 2021, from https://whatis.techtarget.com/definition/reputation-risk

Wu, T., Blackhurst, J., and Chidambaram V. (2005, November 24). *A model for inbound supply risk analysis.* T. Wu et al. / Computers in Industry, 57, 350-365. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.5474&rep=rep1&type=pdf

Appendix 1 - Risk Analysis Procedure Document

Procedure: Risk Analysis Process Document

1.0 SUMMARY

1.1. Project Moksha has established this procedure for leveraging public data of large organizations and their supply chain.

1.2. Responsibility and authority for this procedure are spread across various functions, and defined within this procedure.

1.3. JustProtect is responsible for implementation and management of this procedure.

2.0 REVISION AND APPROVAL

Rev.	Date	Nature of Changes	Approved By
1.0	14 Feb 2021	Original Draft	Von Leue, et al.

3.0 DEFINITIONS (Reorder alphabetically when complete)

3.1. **Client:** The company that has contracted JustProtect to conduct an analysis of its supply chain.

3.2. **Document-Term Matrix:** They are useful in the field of natural language processing and computational text analysis. While the value of the cells is commonly the raw count of a given term, there are various schemes for weighting the raw counts such as relative frequency/proportions and tf-idf.

3.3. **LDA (Latent Dirichlet Allocation):** A generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

3.4. **Library:** The directories where the packages are stored.

3.5. **Python Script:** The python script written by Benjamin Garret and Isabela Rangel. Included in Appendix 1 by reference.

3.6. **R package:** A collection of R functions, compiled code, and sample data. Stored under a directory called “library” in the R environment.

3.7. **Risk**: A negative effect of uncertainty.

3.8. **Risk Analysis Methods:** Strategic tools for understanding market growth or decline, business position, potential and direction for operations. Example tools: PEST, SWOT, Porter's Five Forces Analysis, Competitive Advantage, SOAR, Value Chain Analysis, Core Competencies, Experience Curve Positioning.

3.9. **Risk Assessment:** A systematic investigation and analysis of potential risks, combined with the assignment of severities of probabilities and consequences. These are used to rate risks in order to prioritize the mitigation of high risks.

3.10. **Term:** A word used to describe a thing or to express a concept, especially in a particular kind of language or branch of study.

3.11. **TFIDF (Term Frequency Inverse Document Frequency):** A numerical statistic that is intended to reflect how important a word is to a document. General formula is Times a term appears in a document divided by the number of times that term occurs in all similar documents.

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

Where t denotes the terms; d denotes each document; D denotes the collection of documents.

3.12. **Topic Model:** A type of statistical model for discovering the abstract "topics" that occur in a document.

3.13. **Uncertainty**: A deficiency of information related to understanding or knowledge of an event, its consequence, or likelihood. (Not to be confused with measurement uncertainty.)

3.14.

4.0 PROCEDURE: ANALYSIS OF RISKS

4.1. Risks are analyzed with a focus on determining the likelihood of a given risk existing.

4.2. The overarching view of the process is as follows:

1. Identify a company determined to be significant to the client for this analysis.
2. Publicly accessible filings and documents are acquired in digital form.
3. The text of these forms are passed to the term frequency tools.
4. The output of the tools are sorted to characterize overall risks being discussed.

5. The output of the tools, along with the original text and characterized categories, are used to conduct Text Mining and Topic Modeling.
6. The results of both parts of the analysis are used to concentrate further efforts on researching the identified company's risks.
7. Repeat steps 1-6 for the key suppliers of the company identified in Step 1.
8. Generated a report on this company that presents the data specific to the company.
9. Steps 1-8 are repeated for each company that is determined to be significant to the client for this analysis.
10. Combine the previous analyses to generate a report on this company for presentation to the client.

4.3. Each step of the process is defined in detail in section 5. This document includes the identification and analysis guidelines for key steps associated with the defined process. JustProtect will present the results of the process to the client to address these risks and take action to minimize them.

4.4. The analysis of the companies in the process should aim to answer questions like the following:

What are the opportunities in the market?

What is the competitive landscape in the market?

What are the data regulations that will impact the market?

What are the major growth factors for the regions?

This is a non-exhaustive list, and is merely guidance for the line of thought to be used when conducting the analysis of the automated results.

4.5. Risks will be identified and characterized as part of the end results of the process. This indicates a rough priority, as well as suggesting risk treatment method(s).

4.6. The methods for risk assessments vary, but should always include a means of identifying the risk under examination, and a description of the result of the risk assessment.

4.6.1. Detailed methods may include FMEA (failure mode effects analysis), SWOT (strength, weakness, opportunity and threat) or other tools. No single method is used for all risk assessments; the tool selected should be the best tool applicable to that particular risk analysis.

4.6.2. NOTE: ISO 31010 provides guidance on the selection of risk tools.

4.7. If a risk includes a potential positive aspect, management may elect to conduct an opportunity pursuit assessment on the positive aspect.

5.0 STEP-BY-STEP DESCRIPTIONS

5.1. **Step 1:** Identify a company determined to be significant in the scope of the analysis.

5.1.1. Identify those companies that are within the supply chain of the client, or a selected company. Ideally, this information will be provided by the client.

5.1.2. The identified company(ies) should be scrutinized for relevance to the overall scope of the analysis. Those that are very unlikely to play a current or future role should be considered for excluding from the analysis.

5.2. **Step 2:** Publicly accessible filings and documents are acquired in digital form.

5.2.1. The primary source of this information will be SEC filings, which are accessed via EDGAR.

5.2.2. Other sources will include things such as press releases and news articles.

5.2.3. Due diligence should be exercised to ensure faulty information is not part of these other inclusions.

5.3. **Step 3:** The text of these forms are passed to the python script.

5.3.1. This python script conducts a term frequency-inverse document frequency analysis. This generally identifies which words are being used more often than would generally be expected in a general document.

5.3.1.1. The python script uses the *numpy*, *matplotlib*, *seaborn*, *sklearn, scipy*, and *pandas* libraries, and the *nltk stopwords* corpus to conduct the parsing, cleaning, and analysis.

5.3.2. This will generally involve a simple copy and paste operation. Highlight the text to be used, and paste it into an empty spreadsheet.

5.3.2.1. If the web resource is resistant to copying or selecting, try exploring the page elements/developer view, and/or viewing the source of the page.

5.3.3. The spreadsheet should separate the sentences into separate cells; Split the sentences into separate cells if not.

5.3.4. This document should then be saved as a Comma Separated Values (CSV) file.

5.3.5. Using a python environment, like JuPyterLabs, load the python script. Ensure the csv from the previous step is a part of this environment.

5.3.5.1. For a list of variables in the Python scripts, see Appendix 2.

5.3.6. When run, the python script will generate a csv file. This file [currently] contains entries in 3 columns. The first consists of words it deemed important to the text supplied. The second is a rough estimation of how important the word is. The third contains the sentences that the important word was found in.

5.3.7. The important words file (.\output\[fileprepend]_NLP_importantWords.csv) should be scanned for superfluous results that the python script did not already remove.

5.3.8. These files, in addition to the input file, should then be saved for use with the text mining techniques in Step 4.

5.4. **Step 4:** This sorted data will then be used to undertake a text mining analysis and plot the topic modeling visualizations.

5.4.1. Install necessary R packages and input libraries into RStudio.

5.4.1.1. Packages used: *'tidyverse' , 'tidytext', 'topicmodels', 'igraph', 'ggraph', 'tm'.*

5.4.2. Convert to Document-Term Matrix. This turns the data into a matrix form readable by the '*topicmodels*' package.

5.4.3. Apply LDA.

5.4.4. Approximately determine the appropriate optimal number of topics to be found based on the number of Form 10-K's in the dataset.

5.4.4.1. WILL UPDATE METHOD TO DETERMINE # OF TOPICS FOR WALMART 10-K'S FROM 2011-2020.

5.4.5. Gather the top 10 terms found.

5.4.6. Analyze the association between the set of words for each topic and define what risk the words are most likely to be describing.

5.4.7. Use '*igraph'* and *'ggraph'* for visualization.

5.4.7.1. *The data generated should be formatted in the following way to readily enable company and industry comparisons*: [...]

5.5. **Step 5:** The words passed from the python script should be manually sorted into one of four categories:

5.5.1. Strategic Risks

Risks such as: Regulatory, Technological, Reputational, Demographic trends, Competition, and Capital availability.

5.5.2. Operational Risks

Risks like: Supply chain operations, Human Resources, Information management, business reporting, Planning and Evaluation.

5.5.3. Financial Risks

Common issues: Liquidity, Credit, Inflation, Market performance.

5.5.4. Hazard Risks

Includes risks like: Fire, Theft, Disease and Disability, Liability.

5.6. **Step 6:** The generated results will be used to understand the relative risk levels in comparison to historical measurements and/or industry averages.

5.6.1. Run the time series fraction of the RStudio script.

5.6.1.1. Repeat steps 1 through 5 to obtain the same format of results for at least ten previous 10Ks of the same company to gather sufficient historical information.

5.6.1.2. Place the beta averages per topic in the appropriate location according to the code logic.

5.6.1.3. Execute the chunk of code to obtain the risk levels in comparison with previous years performance.

5.6.2. Run the industry average analysis fraction of the RStudio script.

5.6.2.1. Repeat steps 1 through 5 to obtain the same format of results for multiple companies within the same industry.

5.6.2.2. Group the beta average variables of the different companies under the specified vector according to the topic risk classification.

5.6.2.3. Execute the chunk of code to obtain a grouped bar chart to establish a visual comparison of risks per company.

5.7. **Step 7:** The results of both parts of the analysis are used to concentrate further efforts on researching the identified company's risks.

5.7.1. The charts and graphs generated are assembled into a presentation with descriptive, but non-technical, annotations and captions.

5.7.2. The specific risks should be examined for the availability of more detailed information.

5.8. **Step 8:** This final analysis is used to generate a report on this company.

5.8.1. The uncategorized risks should be contextualized to establish a more detailed understanding of what the risk represents in concrete terms.

5.8.2. The data and graphs for the company should be assembled into an easily referenced report.

5.9. **Step 9:** Repeat steps 1-7 for the key suppliers of the company identified in Step 1.

5.9.1. Applying the 80/20 Rule would be prudent, to avoid overinvesting effort for minimal value add.

5.9.2. Additional depth to the supply chain can be added here, investigating companies that supply the companies identified in Step 1.

5.10. **Step 10:** Steps 1-8 are repeated for each company that is determined to be significant to the client for this analysis.

5.10.1. This can be select competitors, potential supply chain partners, or a general appraisal of a specific industry.

5.11. **Step 11:** Create a comprehensive report that assembles all the data and reports generated into a format that is readily digestible by the client. This will preferably involve creating a dashboard with graphical representations of the overall statistics, as well as links to the individual reports. Tableau or PowerBI are recommended, but not required.

6.0 Appendices

6.1. Appendix 1. Python Scripts

```
"""
# Begin main.py
"""

import NaturalLanguageProcessing
import ScrappedParser
import SecScrapper
import RStudioCSVFormatting
import time


def lookupAndParse(ticker, count):
	ticker = ticker.upper()
	startingTime = time.perf_counter()

	path = SecScrapper.get10kFilings(ticker, count)
	ScrappedParser.processFolder(path, ticker)
	NaturalLanguageProcessing.processAllTicker(ticker)
	RStudioCSVFormatting.processAllTicker(ticker)

	endTime = time.perf_counter() - startingTime
	print("Processing " + str(count) + " " + ticker + " 10-Ks done. " + str(round(endTime, 2)) + "s")



def main():
	SecScrapper.lookUpCiks(40)
	lookupAndParse("TGT", 15)
	#RStudioCSVFormatting.processFile("WMT_0000104169-17-000021_10k_processed")



if __name__ == "__main__":
	main()

"""
# End main.py
# Begin NaturalLanguageProcessing.py
"""

import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
import string
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import Normalizer
from scipy.sparse import csr_matrix
from numpy import array
from numpy import empty
import time
import re
import pathlib
import subprocess
import os
from nltk.corpus import stopwords
from CustomStopwords import my_stopwords


#### config ####
```

```
cleanUp = True
debug = False
################

dir = os.path.dirname(__file__)

''' Goes through input folder and processes everything it can find
'''
def processAllInputFolder():
        processAllTicker("")


def processAllTicker(ticker):
        overallTime = time.perf_counter()
        count = 0
        for fileItem in os.listdir(dir + '/input'):
        if(not fileItem.endswith(".txt")):
        print(fileItem + " is not a .txt file")
        continue
        if(not fileItem.startswith(ticker)):
        continue
        processFile(fileItem)
        count += 1
        subprocess.Popen(r'explorer ' + os.path.abspath(dir + r"/output"))

        finalTime = time.perf_counter() - overallTime
        print("")
        print(" > > > Processed " + str(count) +
        " files. Took " + str(round(finalTime, 2)) + "s")


def processFile(fileItem):

        ''' 1. takes in the file as a parameter
        2. goes through each character in the file
        3. creates a new file that is processed
        '''

        startingTime = time.perf_counter()

        print("---------")
        print('>> Processing: ' + os.path.join(dir,'input', fileItem))

        filename = "/" + fileItem[:-4]

        inputFolder = "/input"
        textFile = open(dir + inputFolder + filename +
        '.txt', 'r', encoding='utf-8')

        lines = textFile.readlines()

        textFileNew = open(dir + inputFolder + filename +
        "_processed" + '.txt', 'w+')

        for line in lines:
        formattedLine = r'{0}'.format(line)

        for character in line:
        letter = r'{0}'.format(character)
        letter = "".join(i for i in letter if ord(i) < 128)
        if(letter.isalnum() or letter.isspace()):
        textFileNew.write(letter)
        elif (letter == "."):
        textFileNew.write('.')
        elif (letter == "-"):
```

```
textFileNew.write(' ')

textFile.close()
textFileNew.close()

read_file = pd.read_csv(
dir + inputFolder + filename + "_processed" + '.txt')
read_file.head(10)
read_file.to_csv(dir + inputFolder + filename +
"_processed" + '.csv', index=None)

# get data out of the table
items = pd.read_csv(dir + inputFolder + filename +
                "_processed" + '.csv', names=["text"])
items.head(10)

items['length'] = items['text'].apply(len)

def text_process(mess):
"""
Takes in a string of text, then performs the following:
1. Remove all punctuation
2. Remove all stopwords
3. Returns a list of the cleaned text
"""
# Check characters to see if they are in punctuation
nopunc = [char for char in mess if char not in string.punctuation]

# Join the characters again to form the string.
nopunc = ''.join(nopunc)

# Now just remove any stopwords
stop_words = set(stopwords.words('english'))
combined_StopWords = stop_words.union(my_stopwords)
return [word for word in nopunc.split() if word.lower() not in combined_StopWords]

bow_transformer = CountVectorizer(analyzer=text_process).fit(items['text'])

items_bow = bow_transformer.transform(items['text'])
sparsity = (100.0 * items_bow.nnz /
(items_bow.shape[0] * items_bow.shape[1]))

tfidf_transformer = TfidfTransformer().fit(items_bow)

# data is transformed, now it needs to be parsed into a datatable
tfidf4 = tfidf_transformer.transform(items_bow)

length = 17150

a = np.empty((length, 3), dtype=object)
b = np.empty(length, dtype=object)
c = np.empty(length, dtype=object)

i = 0
for row in tfidf4:
# gives us an array of split items
wordArray = str(row).split('\n')
for word in wordArray:
if ":" not in word:
if(len(str(word).split(')\t')) <= 1):
i += 1
continue
result = str(word).split(')\t')[1]
score = result
```

```
            wordIndexSplit = str(word).split(')\t')[0].split(', ')[1]
            wordIndex = wordIndexSplit

            # consistency check
            if(i >= len(a)):
            break

            # checks to see if it's numeric, if it is continue loop.
            if(bow_transformer.get_feature_names()[int(wordIndex)].isnumeric()):
            if(debug):
                        print(bow_transformer.get_feature_names()
                        [int(wordIndex)] + " was a number")
            i += 1
            continue

            # add to arrays
            b[i] = bow_transformer.get_feature_names()[int(wordIndex)]

            pattern = str(b[i])

            file = open(dir + inputFolder + filename +
                        "_processed" + '.txt', "r", encoding="utf8")

            result = ""
            for line in file:
            if (re.search(pattern, line) and (len(result) + len(line) < 32000)):
                        result += line
            file.close()

            a[i] = [wordIndex, score, result]
            i += 1

            titles = ["Word Index", "Score", "content"]

            # Display table
            dataframe = pd.DataFrame(data=a, index=b, columns=titles)

            output = dataframe.drop_duplicates(subset=['Word Index', 'Score'], keep='last').sort_values(
            by=['Score']).tail(100).dropna(how='any')
            if(not os.path.isdir(dir + r'/output')):
            os.mkdir(dir + r'/output')
            output.to_csv(dir + "/output" + filename + '_NLP_importantWords.csv')


            # cleanup
            if(cleanUp):
            os.remove(dir + inputFolder + filename + "_processed" + '.txt')
            os.remove(dir + inputFolder + filename + "_processed" + '.csv')

            endTime = time.perf_counter() - startingTime
            print("Processing done. Took " + str(round(endTime, 2)) + "s")

"""
# End NaturalLanguageProcessing.py
# Begin RStudioCSVFormatting.py
"""

import numpy as np
import pandas as pd
import seaborn as sns
from numpy import array
from numpy import empty
import time
import re
import pathlib
import subprocess
import os
import string
from nltk.corpus import stopwords
from CustomStopwords import my_stopwords

''' 1. takes the stopwords from nltk corpus and stores them as a set
        2. unions those stopwords with the custom stopwords that exist in the import CustomStopwords
'''

stop_words = set(stopwords.words('english'))

combined_StopWords = stop_words.union(my_stopwords)

dir = os.path.dirname(__file__)


def processAllInputFolder():
        processAllTicker("")

''' 1. Processes and checks if the file has the extension .txt
        2. Provides the time it took to process the files
'''

def processAllTicker(ticker):
        overallTime = time.perf_counter()
        count = 0
        for fileItem in os.listdir(dir + '/input'):
        if(not fileItem.endswith(".txt")):
        print(fileItem + " is not a .txt file")
        continue
        if(not fileItem.startswith(ticker)):
        continue
        processFile(fileItem)
        count += 1
        subprocess.Popen(r'explorer ' + os.path.abspath(dir + r"/output"))

        finalTime = time.perf_counter() - overallTime
        print("")
        print(" > > > Processed " + str(count) +
        " files into R-studio Format. Took " + str(round(finalTime, 2)) + "s")



def processFile(filename):

        ''' 1. takes the txt file as a parameter and removes the stopwords
        2. goes through each line in the file and checks what section it is in
        3. stores the line and the header scetion it is in into the contentLines list
        4. places everything in a dataframe and replaces any linebreaks and empty space as nan
        5. removes the nan rows due to no data
        6. outputs to a csv file
        '''

        print("---------")
        print('>> Processing: ' + os.path.join(dir,'input', filename))

        filename = filename[:-4]

        textFile = open(dir + "/input/" + filename + ".txt", 'r', encoding='utf-8')
        lines = textFile.readlines()


        def remove_mystopwords(sentence):
        tokens = sentence.split(" ")
        tokens_filtered = [
        word for word in tokens if not word in combined_StopWords]
        return (" ").join(tokens_filtered)


        titles = ["Document", "Paragraph"]
        currentHeader = ""
        contentLines = []

        for line in lines:
        if "item 1a" in line.lower()[:20]:
        currentHeader = filename + "_1A"

        elif "item 7a" in line.lower()[:20]:
        currentHeader = filename + "_7A"

        contentLines.append([currentHeader, remove_mystopwords(line)])
        textFile.close()

        # remember to specify data
        dataframe = pd.DataFrame(data=contentLines, columns=titles).replace(to_replace=["\n"], value=[""], regex=True, inplace=False)
        df1 = dataframe.mask(dataframe['Paragraph'].str.isnumeric()).replace(to_replace="", value=np.nan)
        df = df1.dropna(axis='rows', how="any", subset=["Paragraph"])


        # print(dataframe)
        if(not os.path.isdir(dir + r'/rstudiooutput')):
        os.mkdir(dir + r'/rstudiooutput')

        df.to_csv(dir + "/rstudiooutput/" + filename + "_RStudioFormat.csv", index=False)

"""
# End RStudioCSVFormatting.py
# Begin ScrappedParser.py
"""

from bs4 import BeautifulSoup
import pandas as pd
import re
import os
import time
import unicodedata

'''
Cleans out characters that are not default.
'''
def restore_windows_1252_characters(restore_string):
        """
        Replace C1 control characters in the Unicode string s by the
        characters at the corresponding code points in Windows-1252,
        where possible.
        """

        def to_windows_1252(match):
        try:
        return bytes([ord(match.group(0))]).decode('windows-1252')
        except UnicodeDecodeError:
        # No character at the corresponding code point: remove it.
        return ''

        return re.sub(r'[\u0080-\u0099]', to_windows_1252, restore_string)

'''
Processes all in a single folder path
```

```
'''
def processAllinFolder(folderPath):
        processFolder(folderPath, "")

dir = os.path.dirname(__file__)
''' Processes an entire folder based on the path and starting ticker.
'''
def processFolder(folderPath, ticker):
        #setup a counter
        overallTime = time.perf_counter()
        count = 0
        folderPath = os.path.join(dir, folderPath)

        #Go through each file in the folder and determine if needs processing
        for fileItem in os.listdir(folderPath):
        if(not fileItem.endswith(".txt")):
        print(fileItem + " is not a .txt file")
        continue
        processFile(fileItem, folderPath, ticker)
        count += 1

        finalTime = time.perf_counter() - overallTime
        print("")
        #print counted items
        print(" > > > Parsed " + str(count) +
        " files. Took " + str(round(finalTime, 2)) + "s")

'''
Process a single file based on the fileName, the folder path and the 10K ticker
'''
def processFile(fileItem, folderPath, ticker):

        startingTime = time.perf_counter()

        #remove file extentension
        fileName = fileItem[:-4]
        folderPath = os.path.join(dir, folderPath)

        print("---------")
        print("> Parsing: " + os.path.join(dir, folderPath, fileName + '.txt'))
        textFile = open(os.path.join(dir, folderPath, fileName + '.txt'), 'r', encoding='utf-8')

        soup = BeautifulSoup(textFile, 'lxml')
        textFile.close()

        all_documents = soup.find_all('document')

        #List of desired documents
        document_type_to_get_list = ["10-K", "EX-13"]
        #List of documents to ignore
        document_type_to_ignore = ["GRAPHIC", "XML"]
        #Choice of which list to use. Either "find" or "ignore"
        filter_type = "find"


        text_to_print = ""
        i = 0
        filePrefix = ticker + "_10k_"
        #go through all documents inside the page looking for the specified files.
        #Can either find those by exact title or by ignoring specified titles.
        for document in all_documents:
        #Grab information from the document
        document_type = document.type.find(text=True, recursive=False).strip()

        # Check if it is one of the documents in the list
```

```
if(filter_type == "find"):
terminate = True
for item in document_type_to_get_list:
if item in document_type:
terminate = False
if(terminate):
continue
# Check if the document is not one of the documents to ignore
if(filter_type == "ignore"):
terminate = False
for item in document_type_to_ignore:
if item in document_type:
terminate = True
if(terminate):
continue
# Ensure if document type is not K/A
if(document_type== "10-K/A"):
print("!!! found a 10-K/A, count might be off.")
continue


document_code = document.extract()

if(not document_code):
print("No document code")
continue


document_description = document.description.find(text=True, recursive=False).strip()
document_filename = document.filename.find(text=True, recursive=False).strip()

if(len(document_filename.split(".")) > 1):
filePrefix = document_filename.split(".")[0].upper() + "_"

document_text = document.find('text').extract()

# Break document up into pages
all_thematic_breaks = document_text.find_all('hr')

document_text_string = str(document_text)

all_thematic_breaks = [str(thematic_break)
            for thematic_break in all_thematic_breaks]

regex_delimiter_pattern = '|'.join(map(re.escape, all_thematic_breaks))
split_filing_string = re.split(regex_delimiter_pattern, document_text_string)


# goes through each page that was separated previously
all_text = ""
for page in split_filing_string:
#parse each page
page_soup = BeautifulSoup(page, "html5lib")
page_text = page_soup.html.body.get_text(' ', strip=True)
page_text_norm = restore_windows_1252_characters(
unicodedata.normalize('NFKD', page_text))
#Normalize text and add to the all text string.
page_text_norm = page_text_norm.replace('  ', ' ').replace('\n', ' ')
all_text += page_text_norm
all_text += "\n"
all_text += "\n"

# Go through all text and look for needed sections.
# *Uses item 7A and Market Risks for companies that attach risks in
# the market risks document (EX-13)
```

```
		add_to_file_flag = False
		sentences_added = 0
		character_count = 0
		split_text = all_text.split()
		for index in range(len(split_text)):
		# Checks to see if it is inside an item 7A or 1A and if so then can add to the text
		if(split_text[index].lower() == "item" or split_text[index].lower() == "market" ):
		if(("7a" in split_text[index + 1].lower() or "1a" in split_text[index + 1].lower() or "risk" in split_text[index + 1].lower())):
		add_to_file_flag = True
		sentences_added = 0
		if(("1b" in split_text[index + 1].lower() or "8" in split_text[index + 1].lower())):
		add_to_file_flag = False
		text_to_print += "\n"
		# if inside the item 7A or 1A, proceed
		if(add_to_file_flag):
		if(split_text[index] == "/s/" or split_text[index] == "/"):
		text_to_print += ""
		else:
		if(character_count + len(split_text[index]) > 32767):
			text_to_print += '\n'
			sentences_added = words_added = character_count = 0
		text_to_print += split_text[index]
		text_to_print += " "

		if( split_text[index].endswith(".") and len(split_text[index]) > 4):
		sentences_added += 1


		if(sentences_added % 4 == 3):
		text_to_print += '\n'
		sentences_added = character_count = 0


		#if there was no text, return
		if(text_to_print == ""):
		return

		#If there is not an input path created already, create one
		if(not os.path.isdir(dir + r'/input')):
		os.mkdir(dir + r'/input')

		# Create file and add text to it
		textFile = open(os.path.join(dir + r'/input', ticker + "_" + fileName + '.txt'), 'w+', encoding='utf-8')
		textFile.write(text_to_print)
		textFile.close()

		# print time taken
		endTime = time.perf_counter() - startingTime
		print("Parsing done. Took " + str(round(endTime, 2)) + "s")

"""
# End ScrappedParser.py
# Begin SecScrapper.py
"""

from secedgar.filings import Filing, FilingType
from secedgar.parser import MetaParser
import os
import secedgar

#sets base dir path
dir = os.path.dirname(__file__)
''' Get 10k Filings based on ticker and count
		Returns path to newly scrapped files
'''
```

```
def get10kFilings(companyTicker, count):
        my_filings = Filing(cik_lookup=companyTicker,
                filing_type=FilingType.FILING_10K,
                count=count)
        print(my_filings)
        my_filings.save(os.path.join(dir, 'rawinput/'))
        # path to the scrapped files
        return 'rawinput/' + companyTicker + "/" + "10-k"

''' Looks up CIK dictionary and prints it based on the amount specified
'''
def lookUpCiks(count):
        lookups = dict(list(secedgar.cik_lookup.get_cik_map()["ticker"].items())[:count])
        print(lookups)

"""
# End SecScrapper.py
# Begin CustomStopwords.py
"""

my_stopwords = ['Business', 'Operation', 'Item', '1A', '7A', 'Factors', 'Table of Contents',
        'Walmart', 'EITF', 'FSP', 'SFAS', 'DS', 'Bharti', 'per', 'Seiyu', 'EEOC', 'BCL', 'NYSE', 'Netto', 'CSOP',
        '401k', 'Breyer', 'Odle', 'Sharesave', 'Vips', 'Notes', 'LIFO', 'ASU', 'EMV', 'Yihaodian', 'ASDAs', 'Chile',
        'JD', 'SEC', 'Act', 'Operated', 'Sams', 'Vice', 'President', 'pm', 'Group', 'Asda', 'feet', 'square', 'GILTI',
        'Advent', 'USD', 'calendar', 'Quantitative', 'Qualitative', 'term', 'card', 'Reports', 'CARHCO', 'Sonae',
        'Bompreco', 'McLane', '123R', 'SAB', 'BCL', 'Mr', 'Committee', '141R', 'The', 'the', 'In', 'in', 'ITEM', 'If',
        'if', 'We', 'we', 'may', 'May', 'As', 'as', 'could', 'Could']

"""
# End CustomStopwords.py
"""
```

6.2. Appendix 2. Python Scripts Variables

Main.py		
Line #	Variable	Description
9	ticker	Used to store the ticker that is passed as an argument for the lookupAndParse function and is a string.
10	startingTime	Used to store a performance counter for time and is a float.
17	endTime	Used to store the new performance counter subtracted by the start time to get the time it took to process the 10-Ks.
NaturalLanguageProcessing.py		
Line #	Variable	Description
22	cleanUp	boolean used to delete uneeded files
23	debug	boolean used to print out feature names of the bow_transformer and see if the word was a number
26	dir	string used to store the directory path location to place new folders and files.
35	overallTime	float used to store the performance counter of when this code began running
36	count	int used to store how many files were processed
47	finalTime	float used to store the new performance counter subtracted by the overallTime to get the time it took to process the files
60	startingTime	float used to store the starting performance counter of when the processFile function was used
65	filename	string used to store the fileItem argument
67	inputFolder	string used to store the folder for input
68	textFile	IO used to store the text file to be processed
69	lines	list used to store the lines in the text file
73	textFileNew	IO used to store the processed text file
80	letter	string used to store characters in the line
92	read_file	dataframe used to create the processed csv file
99	items	dataframe used to get data out of the table
113	nopunc	list used to store characters without punctuation
119	stop_words	set used to store stopwords

120	combined_StopWords	set used to store combined stopwords from the custom stopwords library
123	bow_transformer	countvectorizer used to store the countvectorizer of processed text
125	items_bow	csr_matrix used to store the transform of the bow_transformer
129	tfidf_transformer	tfidfTransformer that stores the item_bow
132	tfidf4	object that stores the transform of TfidfTransformer of item_bow
134	length	int used to store a max value used for an array
136	a	ndarray object that stores the context of the wordIndex, score, and result
137	b	ndarray object that stores the feature names by word index
140	i	int used to store a counter used for a consistency check
143	wordArray	list used to store each row in tfidf4 based on line breaks
149	result	list used to store each word in wordArray based on tab stops
150	score	list used to store result
170	pattern	string used to store b as a string
172	file	IO used to store the processed text file
184	titles	list used to store the column titles used for the dataframe
187	dataframe	dataframe used to store the dataframe with the content from a, b, and titles
189	output	dataframe used to store the dataframe with different format
201	endTime	float used to store the new performance counter subtracted by the startingTime
RStudioCSVFormatting		
Line #	Variable	Description
19	stop_words	set used to store stopwords
21	combined_StopWords	set used to store combined stopwords from the CustomStopwords library
23	dir	string used to store the directory path where the script is
34	overallTime	float used to store the performance counter of when the script is being used
35	count	int used to store the number of files processed
46	finalTime	float used to store the new performance counter subtracted by the overallTime
67	filename	string used to store the filename argument from the function processFile
69	textFile	IO used to store the text file that needs to be processed
70	lines	list used to store the lines in the text file
74	tokens	list used to store the sentance based on spaces
75	tokens_filtered	list used to store the words that do not exist in the combined stopwords
80	company_name	string used to store the company name from the text file
81	filing_year	string used to store the filing year from the text file
83	titles	list used to store the column titles used for a dataframe
84	currentHeader	string used to store the header location as the for loop is going through the text file
85	contentLines	list used to store the content for the dataframe
98	dataframe	dataframe used to store a dataframe the contains the contentLines
99	df1	dataframe used to store the dataframe with new format
100	df	dataframe used to store the df1 with new format
ScrappedParser.py		
Line #	Variable	Description
38	overallTime	float used to store the performance counter for when the processFolder function was being used
39	count	int used to store the number of files parsed
40	folderPath	IO used to store the folder path where the script exists
50	finalTime	float used to store the new performance counter subtracted by overallTime
61	startingTime	float used to store the performance counter for when the processFile function was being used
64	fileName	string used to store the name of the file
65	folderPath	IO used to store the folder path where the script exists
69	textFile	IO used to store the text file that needs to be parsed
71	soup	beautifulSoup used to store the text file as an lxml
74	all_documents	ResultSet used to store documents
77	document_type_to_get_list	list used to store the documents needed to get
79	document_type_to_ignore	list used to store the documents not needed
81	filter_type	string used to store the filtering word for the scrappedParser
84	text_to_print	string used to append linebreaks and spaces
91	document_type	object used to store information from the document
95	terminate	boolean used to store the termination value for specific conditions
115	document_code	object used to store the document code number for the document

123	document_filename	object used to store the document filename of the document
128	document_text	object used to store the document text
131	all_thematic_breaks	object used to store all instances of hr thematic breaks
133	document_text_string	string used to store the document text
138	regex_delimiter_pattern	object used to store the delimiter pattern from the thematic breaks
139	split_filing_string	list used to store filing strings
143	all_text	string used to store all text from the split_filing_string
146	page_soup	BeautifulSoup used to store the page in split_filing_string
147	page_text	object used to store the text from the page
148	page_text_norm	string used to store the unicode data for the page
159	add_to_file_flag	boolean used to store the value determining addition to the file
160	sentence_added	int used to store the number of sentences added
161	character_count	int used to store the number of characters added
162	split_text	string used to store split all_text
200	filing_year	string used to store the filing year of the document
208	endTime	float used to store the new performance counter subtracted by startingTime
SecScrapper.py		
Line #	Variable	Description
12	my_filings	Filing used to store the filings
23	lookups	dictionary used to store the cik and ticker of the number of companies specified

6.3. Appendix 3. R Scripts

End Risk Analysis Process Document

Appendix 3 - R scripts

Included by reference in the Risk Analysis Process Document above.

RStudio Script for Topic Modeling Walmart Form 10K: 2019-2021

```
library(data.table)
library(tidyverse)
library(tidytext)
library(topicmodels)
library(readxl)
library(ggplot2)
library(scales)
library(dplyr)
library(readr)
library(tm)
library(pals)
library(RColorBrewer)
library(wordcloud)


# Set Working Directory
setwd("C:/Users/raque/Downloads/PROJECT MOKSHA/15WMTForm10K")

# Import Files : Walmart Form 10K - Items 1A and 7A from the past 3 years >> (2019-2021)
WMTtextdata <- list.files(path = "C:/Users/raque/Downloads/PROJECT MOKSHA/15WMTForm10K",
                       pattern = "*.csv", full.names = TRUE) %>%
lapply(read_csv) %>%
bind_rows


# Convert to Document-Term Matrix
Here we tidy the corpus and apply stopwords.
text_tidy_WMT <- WMTtextdata %>%
  unnest_tokens(word, Paragraph) %>%
  anti_join(stop_words)


# Here we transform the document into a ‘Document-Term Matrix’ which turns the data into a matrix form
readable by the ‘topicmodels’ package.
text_dtm_WMT <- text_tidy_WMT %>%
  count(Document, word, sort = TRUE) %>%
  cast_dtm(Document, word, n)


# Apply LDA.
text_lda_WMT <- LDA(text_dtm_WMT, k = 6, control = list(seed = 1234))
text_lda_WMT
text_topics_WMT <- tidy(text_lda_WMT, matrix = "beta")
text_topics_WMT


# Let’s take a closer look. Here, the top 10 terms are found:
top_terms_WMT <- text_topics_WMT %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
top_terms_WMT

Visualization:
graph_topics_WMT <- top_terms_WMT %>%
  mutate(topic = paste0("Topic ", topic),
         term = reorder(term, beta)) %>%
  ggplot(aes(x = term, y = beta, fill = factor(topic))) +
  geom_col(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~topic, scales = "free_y") +
  coord_flip() +
  labs(x = "", y = expression(beta)) +
  theme_classic()
```

```
graph_topics_WMT

topicToViz <- 5


top40terms <- sort(tmResult$terms[topicToViz,], decreasing=TRUE)[1:40]
words <- names(top40terms)


probabilities <- sort(tmResult$terms[topicToViz,], decreasing=TRUE)[1:40]

mycolors <- brewer.pal(8, "Dark2")
wordcloud(words, probabilities, random.order = FALSE, color = mycolors)
```

~~~
---

title: "R Notebook"

output: html_notebook

---

Import the libraries

```{r}

install.packages("data.table")

library("data.table")

library(tidyverse)

library(tidytext)

library(topicmodels)

library(readxl)

library(ggplot2)

library(scales)

~~~

```{r}

NewWords <- data.frame(c("2009", "2010","2011","2012","2013", "2014", "2015", "2016", "2017","2018",
"2019", "company", "company's", "january", "31", "1","10", "business", "united",

"states", "million", "affect", "u.s", "billion"),

c("SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMAR
T","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART","SMART",

"SMART" ))

names(NewWords) <- c("word", "lexicon")

NewStopWords <- rbind(stop_words, NewWords)

```

Get the csv files.

```{r}

text_Walmart2011 <- read.csv(file = "Walmart2011.csv")

text_Walmart2011$Paragraph <- as.character(text_Walmart2011$Paragraph)

text_Walmart2012 <- read.csv(file = "Walmart2012.csv" )

text_Walmart2012$Paragraph <- as.character(text_Walmart2012$Paragraph)

text_Walmart2013 <- read.csv(file = "Walmart2013.csv")

text_Walmart2013$Paragraph <- as.character(text_Walmart2013$Paragraph)
```

```
text_Walmart2014 <- read.csv(file = "Walmart2014.csv")
text_Walmart2014$Paragraph <- as.character(text_Walmart2014$Paragraph)
text_Walmart2015 <- read.csv(file = "Walmart2015.csv")
text_Walmart2015$Paragraph <- as.character(text_Walmart2015$Paragraph)
text_Walmart2016 <- read.csv(file = "Walmart2016.csv")
text_Walmart2016$Paragraph <- as.character(text_Walmart2016$Paragraph)
text_Walmart2017 <- read.csv(file = "Walmart2017.csv")
text_Walmart2017$Paragraph <- as.character(text_Walmart2017$Paragraph)
text_Walmart2018 <- read.csv(file = "Walmart2018.csv")
text_Walmart2018$Paragraph <- as.character(text_Walmart2018$Paragraph)
text_Walmart2019 <- read.csv(file = "Walmart2019.csv")
text_Walmart2019$Paragraph <- as.character(text_Walmart2019$Paragraph)
text_Walmart2020 <- read.csv(file = "Walmart2020.csv")
text_Walmart2020$Paragraph <- as.character(text_Walmart2020$Paragraph)
text_Walmart2021 <- read.csv(file = "Walmart2021.csv")
text_Walmart2021$Paragraph <- as.character(text_Walmart2021$Paragraph)
```

Topic modeling

```{r}
#2011
topicModelingWalmart11 <- text_Walmart2011 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart11 <- topicModelingWalmart11 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart11 <- LDA(text_dtmWalmart11, k = 4, control = list(seed =
1234))
text_topicsWalmart11 <- tidy(text_ldaWalmart11, matrix = "beta")
top_termsWalmart11 <- text_topicsWalmart11 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2012
topicModelingWalmart12 <- text_Walmart2012 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart12 <- topicModelingWalmart12 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart12 <- LDA(text_dtmWalmart12, k = 4, control = list(seed =
1234))
text_topicsWalmart12 <- tidy(text_ldaWalmart12, matrix = "beta")
```

```
top_termsWalmart12 <- text_topicsWalmart12 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2013
topicModelingWalmart13 <- text_Walmart2013 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart13 <- topicModelingWalmart13 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart13 <- LDA(text_dtmWalmart13, k = 4, control = list(seed =
1234))
text_topicsWalmart13 <- tidy(text_ldaWalmart13, matrix = "beta")
top_termsWalmart13 <- text_topicsWalmart13 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2014
topicModelingWalmart14 <- text_Walmart2014 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart14 <- topicModelingWalmart14 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart14 <- LDA(text_dtmWalmart14, k = 4, control = list(seed =
1234))
text_topicsWalmart14 <- tidy(text_ldaWalmart14, matrix = "beta")
top_termsWalmart14 <- text_topicsWalmart14 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2015
topicModelingWalmart15 <- text_Walmart2015 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart15 <- topicModelingWalmart15 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart15 <- LDA(text_dtmWalmart15, k = 4, control = list(seed =
1234))
```

```
text_topicsWalmart15 <- tidy(text_ldaWalmart15, matrix = "beta")
top_termsWalmart15 <- text_topicsWalmart15 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2016
topicModelingWalmart16 <- text_Walmart2016 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart16 <- topicModelingWalmart16 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart16 <- LDA(text_dtmWalmart16, k = 4, control = list(seed =
1234))
text_topicsWalmart16 <- tidy(text_ldaWalmart16, matrix = "beta")
top_termsWalmart16 <- text_topicsWalmart16 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2017
topicModelingWalmart17 <- text_Walmart2017 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart17 <- topicModelingWalmart17 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart17 <- LDA(text_dtmWalmart17, k = 4, control = list(seed =
1234))
text_topicsWalmart17 <- tidy(text_ldaWalmart17, matrix = "beta")
top_termsWalmart17 <- text_topicsWalmart17 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2018
topicModelingWalmart18 <- text_Walmart2018 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart18 <- topicModelingWalmart18 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart18 <- LDA(text_dtmWalmart18, k = 4, control = list(seed =
1234))
text_topicsWalmart18 <- tidy(text_ldaWalmart18, matrix = "beta")
top_termsWalmart18 <- text_topicsWalmart18 %>%
 group_by(topic) %>%
```

```
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2019
topicModelingWalmart19 <- text_Walmart2019 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart19 <- topicModelingWalmart19 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart19 <- LDA(text_dtmWalmart19, k = 4, control = list(seed =
1234))
text_topicsWalmart19 <- tidy(text_ldaWalmart19, matrix = "beta")
top_termsWalmart19 <- text_topicsWalmart19 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2020
topicModelingWalmart20 <- text_Walmart2020 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart20 <- topicModelingWalmart20 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart20 <- LDA(text_dtmWalmart20, k = 4, control = list(seed =
1234))
text_topicsWalmart20 <- tidy(text_ldaWalmart20, matrix = "beta")
top_termsWalmart20 <- text_topicsWalmart20 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
#2021
topicModelingWalmart21 <- text_Walmart2021 %>%
 unnest_tokens(word, Paragraph) %>%
 anti_join(NewStopWords)
text_dtmWalmart21 <- topicModelingWalmart21 %>%
 count(Document, word, sort = TRUE) %>%
 cast_dtm(Document, word, n)
text_ldaWalmart21 <- LDA(text_dtmWalmart21, k = 4, control = list(seed =
1234))
text_topicsWalmart21 <- tidy(text_ldaWalmart21, matrix = "beta")
top_termsWalmart21 <- text_topicsWalmart21 %>%
 group_by(topic) %>%
 top_n(12) %>%
 ungroup() %>%
 arrange(topic, -beta)
```
~~~
Beta Average 2011
```{r}
topic1Walmart11 <- top_termsWalmart11 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart11 <-mean(topic1Walmart11$beta)
topic2Walmart11 <- top_termsWalmart11 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart11 <- mean(topic2Walmart11$beta)
topic3Walmart11 <- top_termsWalmart11 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart11 <- mean(topic3Walmart11$beta)
topic4Wlamart11 <- top_termsWalmart11%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart11 <-mean(topic4Wlamart11$beta)
~~~
~~~
Beta Average 12
```{r}
topic1Walmart12 <- top_termsWalmart12 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart12 <-mean(topic1Walmart12$beta)
topic2Walmart12 <- top_termsWalmart12 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart12 <- mean(topic2Walmart12$beta)
topic3Walmart12 <- top_termsWalmart12 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart12 <- mean(topic3Walmart12$beta)
topic4Wlamart12 <- top_termsWalmart12%>%
~~~

~~~
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart12 <-mean(topic4Wlamart12$beta)
```
Beta Average 13
```{r}
topic1Walmart13 <- top_termsWalmart13 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart13 <-mean(topic1Walmart13$beta)
topic2Walmart13 <- top_termsWalmart13 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart13 <- mean(topic2Walmart13$beta)
topic3Walmart13 <- top_termsWalmart13 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart13 <- mean(topic3Walmart13$beta)
topic4Wlamart13 <- top_termsWalmart13%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart13 <-mean(topic4Wlamart13$beta)
```
Beta Average 14
```{r}
topic1Walmart14 <- top_termsWalmart14 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart14 <-mean(topic1Walmart14$beta)
topic2Walmart14 <- top_termsWalmart14 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart14 <- mean(topic2Walmart14$beta)
topic3Walmart14 <- top_termsWalmart14 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart14 <- mean(topic3Walmart14$beta)
topic4Wlamart14 <- top_termsWalmart14%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart14 <-mean(topic4Wlamart14$beta)
```
Beta Average 15
```{r}
~~~

```
topic1Walmart15 <- top_termsWalmart15 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart15 <-mean(topic1Walmart15$beta)
topic2Walmart15 <- top_termsWalmart15 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart15 <- mean(topic2Walmart15$beta)
topic3Walmart15 <- top_termsWalmart15 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart15 <- mean(topic3Walmart15$beta)
topic4Wlamart15 <- top_termsWalmart15%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart15 <-mean(topic4Wlamart15$beta)
```

```
Beta Average 16
```

```{r}
topic1Walmart16 <- top_termsWalmart16 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart16 <-mean(topic1Walmart16$beta)
topic2Walmart16 <- top_termsWalmart16 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart16 <- mean(topic2Walmart16$beta)
topic3Walmart16 <- top_termsWalmart16 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart16 <- mean(topic3Walmart16$beta)
topic4Wlamart16 <- top_termsWalmart16%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart16 <-mean(topic4Wlamart16$beta)
```

```
Beta Average 17
```

```{r}
topic1Walmart17 <- top_termsWalmart17 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart17 <-mean(topic1Walmart17$beta)
topic2Walmart17 <- top_termsWalmart17 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart17 <- mean(topic2Walmart17$beta)
topic3Walmart17 <- top_termsWalmart17 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart17 <- mean(topic3Walmart17$beta)
topic4Wlamart17 <- top_termsWalmart17%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart17 <-mean(topic4Wlamart17$beta)
```
Beta Average 18
```{r}
topic1Walmart18 <- top_termsWalmart18 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart18 <-mean(topic1Walmart18$beta)
topic2Walmart18 <- top_termsWalmart18 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart18 <- mean(topic2Walmart18$beta)
topic3Walmart18 <- top_termsWalmart18 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart18 <- mean(topic3Walmart18$beta)
topic4Wlamart18 <- top_termsWalmart18%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart18 <-mean(topic4Wlamart18$beta)
```
Beta Average 19
```{r}
topic1Walmart19 <- top_termsWalmart19 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart19 <-mean(topic1Walmart19$beta)
topic2Walmart19 <- top_termsWalmart19 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart19 <- mean(topic2Walmart19$beta)
topic3Walmart19 <- top_termsWalmart19 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart19 <- mean(topic3Walmart19$beta)
```

~~~
topic4Wlamart19 <- top_termsWalmart19%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart19 <-mean(topic4Wlamart19$beta)
```
Beta Average 20
```{r}
topic1Walmart20 <- top_termsWalmart20 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart20 <-mean(topic1Walmart20$beta)
topic2Walmart20 <- top_termsWalmart20 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart20 <- mean(topic2Walmart20$beta)
topic3Walmart20 <- top_termsWalmart20 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart20 <- mean(topic3Walmart20$beta)
topic4Wlamart20 <- top_termsWalmart20%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart20 <-mean(topic4Wlamart20$beta)
```
Beta Average 21
```{r}
topic1Walmart21 <- top_termsWalmart21 %>%
 filter(topic == 1) %>%
 select(beta)
meanTopic1Walmart21 <-mean(topic1Walmart21$beta)
topic2Walmart21 <- top_termsWalmart21 %>%
 filter(topic == 2) %>%
 select(beta)
meanTopic2Walmart21 <- mean(topic2Walmart21$beta)
topic3Walmart21 <- top_termsWalmart21 %>%
 filter(topic == 3) %>%
 select(beta)
meanTopic3Walmart21 <- mean(topic3Walmart21$beta)
topic4Wlamart21 <- top_termsWalmart21%>%
 filter(topic == 4) %>%
 select(beta)
meanTopic4Walmart21 <-mean(topic4Wlamart21$beta)
```
Topics Modeling Graphs
~~~

~~~
```{r}
graph_topicsWalmart11 <- top_termsWalmart11 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart11
graph_topicsWalmart12 <- top_termsWalmart12 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart12
graph_topicsWalmart13 <- top_termsWalmart13 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart13
graph_topicsWalmart14 <- top_termsWalmart14 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart14
graph_topicsWalmart15 <- top_termsWalmart15 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
~~~

```
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart15
graph_topicsWalmart16 <- top_termsWalmart16 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart16
graph_topicsWalmart17 <- top_termsWalmart17 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart17
graph_topicsWalmart18 <- top_termsWalmart18 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart18
graph_topicsWalmart19 <- top_termsWalmart19 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart19
graph_topicsWalmart20 <- top_termsWalmart20 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart20
graph_topicsWalmart21 <- top_termsWalmart21 %>%
 mutate(topic = paste0("Topic", topic),
        term = reorder(term, beta)) %>%
 ggplot(aes(x = term, y = beta, fill = factor(topic))) +
 geom_col(alpha = 0.7, show.legend = FALSE) +
 facet_wrap(~topic, scales = "free_y") +
 coord_flip() +
 theme_minimal() +
 labs(x = "", y = expression(beta))
graph_topicsWalmart21
```

DataFrame Values

```{r}
FinancialRisk <-
c(meanTopic3Walmart11,meanTopic2Walmart12,meanTopic3Walmart13,
meanTopic1Walmart14, meanTopic1Walmart15, meanTopic1Walmart16,
meanTopic2Walmart17, meanTopic3Walmart18, meanTopic3Walmart19,
meanTopic4Walmart20, meanTopic3Walmart21)
OperationalRisk <- c(meanTopic1Walmart11, meanTopic1Walmart12,
meanTopic2Walmart13, meanTopic2Walmart14, meanTopic3Walmart15,
meanTopic3Walmart16, meanTopic4Walmart17, meanTopic2Walmart18,
meanTopic1Walmart19, meanTopic3Walmart20, meanTopic1Walmart21)
RegulatoryRisk <- c(meanTopic2Walmart11, meanTopic3Walmart12,
meanTopic4Walmart13, meanTopic4Walmart14, meanTopic2Walmart15,
meanTopic2Walmart16, meanTopic3Walmart17, meanTopic1Walmart18,
meanTopic4Walmart19, meanTopic1Walmart20, meanTopic2Walmart21)
year <- c("2011", "2012", "2013","2014", "2015", "2016","2017", "2018",
"2019","2020", "2021")
DataFrameRisks <- data.frame(FinancialRisk, OperationalRisk,
RegulatoryRisk, year)
```

~~~
TimeSeries Graph
```{r}
ggplot(data = DataFrameRisks, aes(x = year, y = FinancialRisk, group = 1))
+
 geom_line(color = 'red', alpha = 0.45, size = 1.7) +
 geom_line(y = RegulatoryRisk, group = 1, color = 'blue', alpha = 0.45,
size = 1.7)+
 geom_line(y = OperationalRisk, group = 1, color = 'green', alpha = 0.45,
size = 1.7)+
 ylab("Risk Intensity based on Beta Average") +
 ggtitle("Time Series Walmart Risk Development from 2011 to 2021")+
 theme_light()
``
~~~