Master Thesis Econometrics and Management Science

Quantitative Marketing & Business Analytics

Identifying Takeover Targets using Text Analytics

Author:
Tim de Jager
Studentnumber:
408569

Supervisor:
Dr. M van de Velden
Second assessor:
Prof. dr. DJC van Dijk

November 13, 2020

Abstract

Mergers and acquisitions involving publicly traded companies are of interest to investors as stock prices of companies that are acquired by another company rise over 30% on average after the transaction is announced. Therefore, I study different models to identify takeover targets using text analytics. The study is conducted on two sections from 10-K filings. 10-K filings are reports that publicly traded companies are obliged to file to the Security Exchange Commission annually. Using several combinations of document vectorization methods and classification methods, portfolios are created and analyzed based on the percentage of correctly classified takeover targets in a portfolio and the market performance of a portfolio. The best performing portfolio finds that 7.8% (230 out of 2945) of the companies in the portfolio are takeover targets, it identifies 18.1% (230 out of 1271) of all the takeover targets in the data to be in the portfolio, and yields an annual abnormal return of more than 9%. Furthermore, I find that classification models that identify takeover targets based on text analysis do not outperform prediction models based on numerical characteristics in terms of the percentage of identified takeover targets and the market performance.

Contents

1 Introduction and literature review

Predicting mergers and acquisitions (M&A) using statistical models is an area of great interest in finance. Investors build prediction models that predict M&A due to the fact that acquirers usually pay a premium for the acquisition of shares of the target company during these M&A transactions. This acquisition premium, also called the goodwill, can be paid for many reasons. Acquirers pay an acquisition premium as the acquirer believes the synergy creates more value than the cost of the acquisition. From a target perspective, the current shareholders of the takeover target expect a premium for their shares as they want to benefit optimally. This means that if one can effectively create an investment portfolio of companies that will be acquired or merged in the period the companies are in the portfolio, referred to as the takeover targets, investing in this investment portfolio likely results in positive abnormal returns, also referred to as positive alpha's(α). The abnormal return is created by the acquisition premium paid to the shareholders of the target (Franks and Harris, 1989). Early studies based on predicting takeover targets use only numerical characteristics of companies, like financial values and ratios, focus on predictive power of companies in the sample only, see for example Stevens (1973) and Dietrich and Sorensen (1984). Next to that, no investment portfolios with positive abnormal returns are created in these studies. Palepu (1986) indicates that the prediction accuracy of targets in non-random samples, from Stevens (1973) and Dietrich and Sorensen (1984), generate inconsistent and biased estimations. He finds that the predictive power of his model, that adopts for methodological modifications to avoid the problems in earlier research, results in unbiased and consistent estimates. His model invests in 625 companies and does not yield excess returns, and classifying takeover targets in the investment portfolio comes at the cost of classifying a high number of non-takeover target firms in the portfolio too.

In this thesis, I try to bridge the gap between identifying takeover targets based on numerical characteristics of companies and analysing textual data. To the best of my knowledge, no study has been conducted on identifying takeover targets using the textual data, and building investment portfolios accordingly.

In the corporate world, around 80% of the data available is textual data (Ur-Rahman and Harding, 2012). Textual data is available in different formats that can be analyzed, like corporate reports, newspaper items and online articles. In the United States, all companies that are publicly traded are obligated to file different types of corporate filings to the U.S. Security Exchange Commission (SEC). Publicly traded companies are obliged to file 10-K filings (10-Ks) and 10-Q filings (10-Qs)

to the SEC. A 10-K is an annual audited comprehensive overview of the company's operations, financial status, risks, operating market and more, and a 10-Q is a quarterly truncated version of the 10-K that includes important updates for the firm during the business year. The 10-Ks are more extensive than the firms annual reports and are usually more than 100 pages long. Therefore, reading and analyzing these 10-Ks can give a clear insight in the state of the business. 10-Ks are found in the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database from the SEC, which is publicly available online. In the Appendix, additional historical background about the SEC, EDGAR and 10-Ks can be found. Cohen et al. (2016) study the changes in 10-Ks and 10-Qs thoroughly. They show that changes made to the 10-Ks in two particular sections of the document, the Management's Discussion and Analysis (MD&A) and Risk Factors (RF) sections, are informative about future returns. A portfolio that shorts companies that changes the RF of their 10-Ks and 10-Qs and buys companies that keep this segment almost identical earns over 22% annually by means of the excess return in the Fama-French 5-factor model. Furthermore, they find proof for Brown and Tucker (2011)'s finding who state that MD&A section growth and similarity over time point to less usefulness of the MD&A section when looking to future returns is not true in the sense that the textual content of the 10-Ks are useful to gain abnormal returns. Therefore, in this thesis, I analyze two sections of the 10-Ks, the MD&A and RF sections, based on their textual content. From all 10-Ks filed to the SEC in EDGAR between 2005 and 2018, in this thesis, 46,124 and 43,948 10-Ks are used with the MD&A and RF sections respectively.

Due to great improvements in computing power, text analytics is used throughout numerous disciplines nowadays (Loughran and McDonald, 2016). In finance and accounting, 10-Ks are widely accepted as informative documents that provide clear business insights. Li (2008) is one of the first to analyze 10-Ks using text analysis based on readability. He investigates mainly 10-Ks filed between 1993 and 2003, and finds that firms with lower earnings are harder to read based on the Fog-index and length compared to firms with higher earnings. The Fog-index is a readability test for English written texts to provide estimates of the number of educational years one needs to have had to be able to read the text. In a sentiment analysis based on financial news, Yazdani et al. (2017) finds that term frequency-inverse document frequency (tf-idf) feature weighing combined with both unigrams and bigrams perform best using support vector machines (SVM) with the radial basis function (RBF) kernel. Li (2010) uses sentiment analysis and finds that the tone in the Forward Looking Statement (part of the MD&A section in 10-Ks) is positively related (on average) to future earnings. Based on 10-Ks, Loughran and McDonald (2011) create 7 word list categories

(Negative, Positive, Uncertainty, Litigious, Strong Modal, Weak Modal, Constraining) for financial documents. They show that these word lists are better suited for sentiment analysis in financial texts than earlier used word lists such as the Harvard IV-4 dictionaries[1]. The word lists are created using words from all 10-K filings available between 1994 and 2008. They use these word lists to measure tone and sentiment in 10-Ks and to analyze the effect of stock returns. They provide evidence that some of the word lists are linked to trading volumes, volatility and earnings around the date of filing. Next to that, Li (2010) conducted research on a particular part of the MD&A section, the Forward Looking Statement, in the 10-Ks. He finds a positive tone (a high amount of 'positive' words in a text document) is positively associated with future earnings. In the MD&A section, a company's management discusses the prior, current and future state of the company in a more detailed manner. This part of the 10-K is usually written by the management in their own words. The fact this MD&A section is written in their own words is of importance for the insights one wants to get out of textual data. Furthermore, the MD&A section gives a broader insight in management's beliefs of the company, the business risks. Even the material changes, expectations and assumptions about the future of the company can be included in the MD&A. The MD&A section is most used in the literature with regard to text analysis of 10-Ks. As 10-Ks are regarded as informative, using document vectorization methods to create numerical vectors combined with classification algorithms could result in creating portfolios that have a high percentage of takeover targets and portfolios that outperform the markets.

Therefore, I use three document vectorization techniques (Bag of Words, Term Frequency - Inverse Document Frequency and Doc2Vec) to vectorize the textual content in the MD&A and RF sections to numerical vectors which can be used in combination with classification algorithms. Based on the created document vectors, three classification algorithms (logistic regression, multinomial naive Bayes and support vector machines) are used to create probabilities for 10-Ks filed from 2010 to 2018. These probabilities are created based on the filings filed in the 5 previous years, which are used as the training data for every year from 2010 to 2018. Then, portfolios are created using the 10% most likely companies that are takeover targets within one year after the 10-K is filed for every year from 2010 to 2018. These portfolios are then analyzed using two indications of a good performing portfolio: (1) a high percentage of takeover targets in the portfolio and (2) the portfolio outperforming the market.

All 10-Ks combined create a large set of documents. Vectorizing these documents combined

[1]See `http://www.wjh.harvard.edu/~inquirer/homecat.htm`

with classification methods can be a powerful method to classify and identify takeover targets. Therefore, the first research question is:

> *1. Which combination of document vectorization techniques based on the textual data in 10-Ks and classification methods of these document vectors, if any, is able to best identify takeover targets in the portfolio?*

Furthermore, one of the goals of identifying takeover targets is to create a portfolio that outperforms the market index with positive abnormal returns (alpha's). Portfolios created using models with a high percentage of identified takeover targets are more likely to generate positive alpha's due to the acquisition premium paid by the acquiring company. Therefore, the next question is:

> *2. Which portfolio, if any, based on document vectorization techniques and classification methods is best to invest in with regard to outperforming the market and the percentage of takeover targets in the portfolio?*

The best performing portfolio finds that 7.8% (230 out of 2945) of the companies in the portfolio are takeover targets, the portfolio finds 18.1% (230 out of 1271) of all the actual takeover targets in the data to be identified in the portfolio, and yields an annual abnormal return of more than 9%.

Next to that, I am interesting in the performance of the text based classification models compared to classification models based on numerical characteristics of companies. Therefore, I compare results from the literature to results in this study.

Morck et al. (1988) note that a clear distinction must be made in the classification of friendly (synergistic) and hostile (disciplinary) takeovers. They state that takeover targets all belong to the same group, the takeover targets, though the dissimilarities between hostile and friendly takeovers are not taken into account. Powell (2004) proposes several takeover prediction models based on the distinction made in Morck et al. (1988). Powell (2004) finds that multinomial models using two takeover groups, hostile and friendly, increase the explanatory power of the percentage of predicted takeover targets compared to binomial models using one group of takeovers. Furthermore, his multinomial models create a buy-and-hold strategy with significant positive abnormal returns when the strategy is to invest in predicted hostile targets only. Three portfolios constructed create a percentage of identified takeover targets of more than 5%. However, these portfolios do consist of only 15, 7 and 3 stocks to invest in. He also finds that misclassified predicted hostile takeover targets generate positive abnormal returns. Therefore, he concludes that investing in companies that are depicted as hostile takeovers are more profitable.

Based on a sample of listed firms in Australia, Rodrigues and Stevenson (2013) find that a combined forecasting model based on logistic and neural network models outperforms the individual models. They state that the combined forecasting method should be used to improve takeover target prediction accuracy. Their out-of-sample portfolios created by the combined forecasting models correctly classify 3 of the 19 (15.79%) classified targets as takeover target in 2009, 9 of the 40 (22.50%) classified targets as takeover target in 2010, and 6 of the 18 (33.33%) classified targets as takeover target in 2011. The investment portfolio created using these classified takeover targets generated no abnormal return in 2009. However, in 2010 and 2011 the portfolios represented an abnormal return of 5.45% and 6.78% respectively.

Using a sample of stocks and companies trading on the London Stock Exchange from July 1988 to June 2011, Danbolt et al. (2016) create several portfolios based on logit models with different variables. The takeover target portfolios are created based on the 20% firms with the highest takeover probability (top quintile function) per year 1995 to 2009, which is their 15-year holdout period. Their conventional model based on eight hypotheses in the literature performs best as 302 of the 3545 (8.52%) predicted takeover targets are actual takeover targets. Investing in the portfolio created using the conventional model do not results in abnormal returns. However, as they find that target firms and distressed firms have common firm characteristics, mitigating possible bankrupt firms from the portfolio results in a portfolio with abnormal returns of 0.9% per month (10.8% annually) based on the Fama-French Three-Factor Model.

Comparing the results in this study with the literature, results in the third and last research question:

> *3. Do classification models based on textual data from 10-Ks perform better than classification models based on numerical characteristics in terms of the percentage of takeover targets in the portfolio and outperforming the market?*

In this thesis, I find that the classification models used to identify takeover targets based on 10-Ks generate equal results compared to classification models based on numerical characteristics from the literature in terms of the percentage of takeover targets in the portfolio and the market performance of the portfolio.

This thesis is further organized as follows. Section 2 describes the data collection process The methods used to vectorize and classify the documents, and the models to evaluate the portfolios are described in Section 3. Section 4 shows the results of the created portfolios. Section 5 concludes

this thesis by answering the research questions.

2 Data

2.1 10-K scraping

While the MD&A section has been part of the 10-K for a long time, the RF section has only been introduced since the start of 2005. Therefore, all 10-K filings filed by publicly traded companies on either the New York Stock Exchange (NYSE), the New York Stock Exchange American (AMEX) and the Nasdaq Stock Market (Nasdaq) between 2005 and 2018 are extracted. Daily and monthly stock data is available through the Center for Research in Security Prices (CRSP) database until the 31st of December 2019, which means the latest possible filing date would be the 31st of December 2018 for a one year gap. Therefore, the end of this time frame is chosen in such a way that for every 10-K filed by a company, it is known whether a firm was acquired within a period of one year after the 10-K was filed.

The complete list, including the URL links to the 10-K submissions, of all filings filed to the SEC are stored in the online Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database. There are more than 117,000 10-K filings stored in EDGAR between 2005 and 2018. Every company has one observation per year if a 10-K is filed. As the file size of a 10-K document is usually more than 10 megabytes, downloading all these documents in the period between 2005 and 2018 is not possible due to limited storage. Therefore, only the relevant information, the MD&A and RF sections, is extracted from the 10-Ks.

Information about all completed M&A transactions from 2005 to 2019 are retrieved from the Securities Data Company (SDC) Platinum database.

Complete so-called *master* dictionaries including the URL to a text file with HTML code of a full disclosure with additional information filed to the SEC by companies in any quarter of any year from 1993 until today can be retrieved from the EDGAR database archives [2]. From the EDGAR archives, the following data is obtained for every disclosure filed in a quarter of a given year:

- *Central Index Key (CIK)*: a unique number consisting of 10 digits for every company that needs to file disclosures to the SEC.
- *Company Name*: the name of the company as stated in the EDGAR database.

[2] `https://www.sec.gov/Archives/edgar/full-index/`

- *Form Type*: type of the form.
- *Date Filed*: the date at which the file was sent to the SEC.
- *Filename*: final part of the URL to the text file of HTML code of the filing available in the EDGAR database, add `https://sec.gov/archives/` at the start.

In total, the number of 10-Ks in the database from 2005 until the 31st of December 2018 is 117,529. Several filters and requirements are made previous to downloading the full textual data of the 10-Ks and extracting the right sections. Table 1 shows the effect on the sample size for all filters and requirements in the data. The motivation for the filters and requirements are also briefly described below.

2.1.1 Motivation for Filters and Requirements

The first filing of every CIK for every year is taken and there needs to be a minimum of 180 days between the filings year on year. For every 10-K, I need to have unique identifiers to be able to merge the filings with takeover data. Therefore, I require the CIK of that filing to match with the CRSP/Compustat Merged (CMM) database available through the Wharton Research Data Services (WRDS) to get the CRSP's PERMNO identifiers, the Compustat's GVKEY identifier and SDC Platinum's CUSIP code matching with the CIK. A PERMNO is a permanent unique identifier for a security, a GVKEY is a company unique identifier and a CUSIP is a unique identifier for North American Securities. To identify whether a company was on the Nasdaq, NYSE or AMEX, I use the CRSP database which provides historical prices, volumes traded, shares outstanding, exchanges it trades on and further information of securities for every trading day a security is trading on the stock market. The portfolio is updated monthly and stocks of identified companies are in the portfolio for 12 months. Therefore, the last trading day of every filing date's month is chosen as the first investment or buy date. This means that a company that files their 10-K on 15 September 2005 is added to the portfolio only on the last trading day of September 2005, which is 30 September 2005 and is in the portfolio until August 2006. Therefore, it is required for every 10-K to trade on the potential buy date.

To match the 10-Ks to the M&A transactions, the SDC Platinum database is used to retrieve all transactions completed for which a company was acquired or merged while trading on the Nasdaq, NYSE or AMEX and the announcement date of the takeover was between the 1st of January 2005

[3] 4 transactions are linked with 2 10-Ks

Table 1: Configuration of the 10-K sample used.

Source/Filter	Sample Size 10-Ks	Observations Removed	Matched takeovers
10-K documents			
10-K 2005Q1-2018Q4 total sample	117,529		
First filing per CIK per year	115,052	2477	
Minimum of 180 days between firm's annual 10-Ks	114,815	237	
CMM, CRSP and SDC Platinum matching			
CMM database match	58,848	55,967	
Trading on potential buy date	57,765	1083	
SCD Platinum's M&A transactions match	57,769[3]		2579
Filters on buy date			
Price and shares outstanding information	57,755	14	2579
Listed on Nasdaq, NYSE or AMEX	57,712	43	2579
Stock price $\geq$ \$1	56,542	1170	2490
Market capitalization $\geq \$50,000,000$	50,366	6176	2227
MD&A section			
MD&A section identified	49,783	583	2194
MD&A section $\geq$ 250 words	46,124	3579	2072
Risk Factors section			
Risk Factors section identified	45,615	4751	1989
Risk Factors section $\geq$ 250 words	43,948	1645	1929

and the 31st of December 2019. Transactions are linked to a 10-K if the announcement date of the transaction falls between the filing date of the 10-K and 251 trading days, equal to almost exactly one year, after the filing date.

After matching the different databases and sources, all matched 10-Ks are required to contain the daily closing price and information on the total number of shares outstanding on the last trading day of the filing date's month as this is used for the calculation of the total market capitalization.

To reduce noise in the data from small companies, companies that are referred to as penny

stocks or that have a small market capitalization are removed. Penny stocks are defined by the SEC as stocks that are trading for less than \$5. As this includes a lot of stocks of companies with high market capitalization it is more common for investors to refer to penny stocks as stocks trading for less than \$1. Companies with a market capitalization of less than $\$50,000,000$ are referred to as nano capitalization companies and are also removed. Alteryx is used to merge the different data sets, to remove the unwanted 10-Ks and to produce the final data set of 10-Ks of which the MD&A and RF sections are extracted.

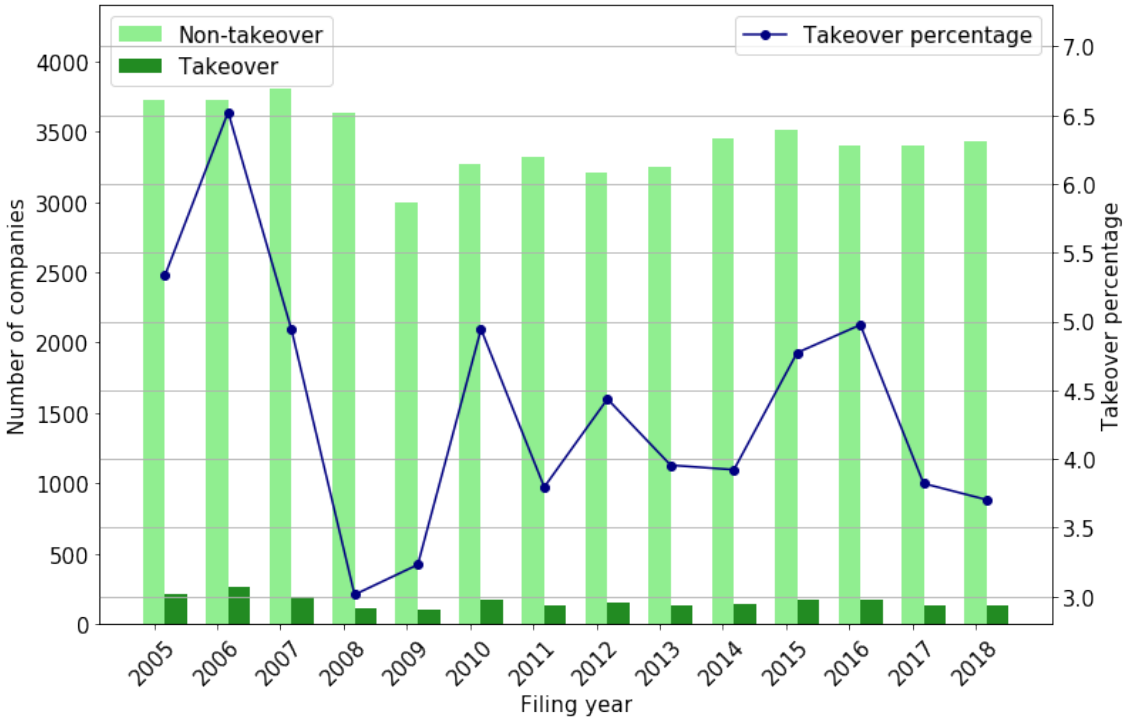

Figure 1: The number of companies that were non-takeover and takeover target in the year of the filing date of the 10-Ks in the sample.

In total, 50,366 companies with 2227 takeovers are used for the actual textual data extraction from the EDGAR database. Figure 1 shows the number of companies that were takeover targets and non-takeover targets, and the percentage of companies taken over in this sample. On average 4.3% of the companies were acquired annually. Figure 2 displays the average annual return for both the non-takeover and the takeover targets including the percentage difference between a portfolio with all takeover targets and with all non-takeover targets. Figure 2 clearly shows the higher returns for the takeover targets. An investment portfolio consisting of all takeover targets in our sample would have an annual return of 35.1% in total. However, the non-takeover targets only have 7.7%

annual return. This shows that investing in takeover targets results in positive abnormal returns. Therefore, identifying companies that are takeover targets might lead to positive abnormal returns when enough takeover targets are in the investment portfolio.

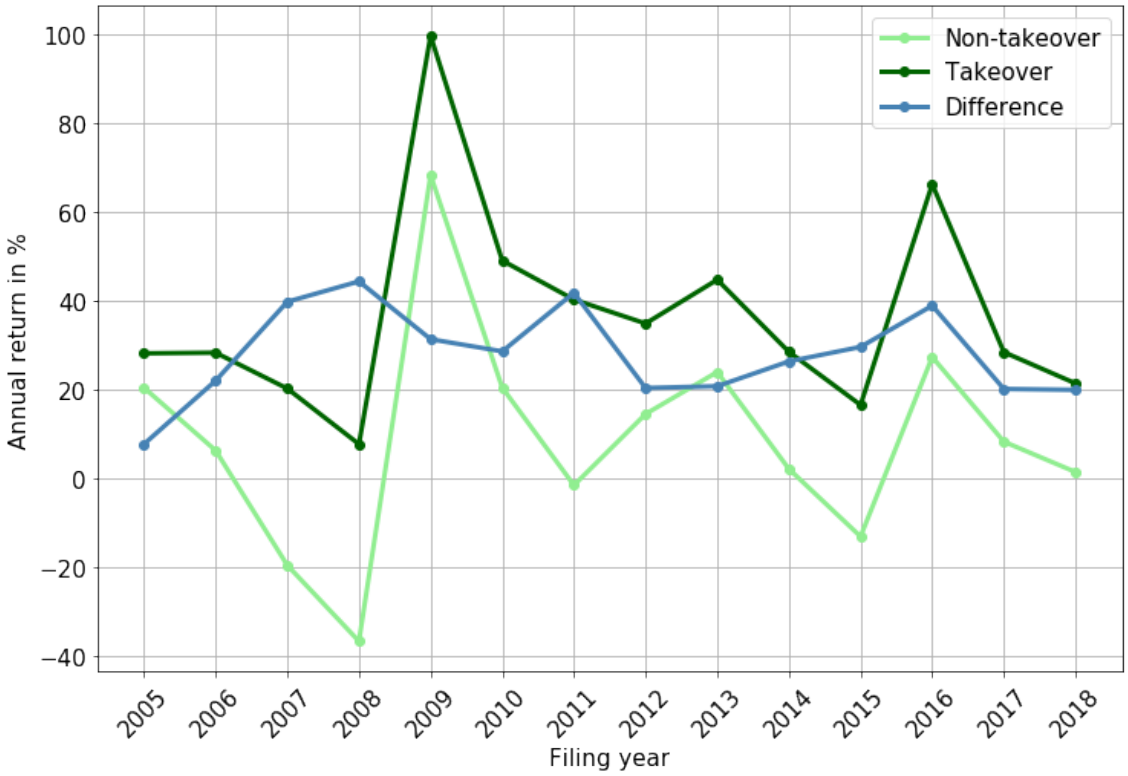

Figure 2: The average annual return of non-takeover and takeover targets, and the difference in the average return between the two classes.

2.2 The MD&A and Risk Factors section extraction

The 50,366 10-Ks are downloaded from the EDGAR database using the URL to the text file of the actual 10-Ks. These URLs refer to a text file which includes HTML code of the actually filed filing, exhibits, graphics, PNG files, PDF files, Excel files and XBRL files (used for financial information) filed alongside the textual content of the filing. As I only focus on the textual content of the filing, I download text of every filing and I clean the HTML code using regular expressions to preserve only the textual content of the filing and to eliminate useless parts of the text file. Regular expressions are a technique to find and replace certain patterns in strings. In this case, regular expressions are used as there is no clear structure in the filings. Though there is structure in the textual content due to the different parts and items, see Table 6. I partly rely on the approach by Loughran and

McDonald (2011)[4] to extract only the textual content from the filings.

To clean the textual data, unnecessary parts are removed from the HTML code, including exhibits, graphics, PDF-file, brackets, tags, HTML unicode and tables. After cleaning the downloaded text files, only the RF and MD&A section need to be retrieved as the files are large and storage is limited. By identifying the placement of the items in the 10-Ks, the sections are retrieved most of the time. The MD&A section was not identified in only 583 cases (just over 1%) due to the MD&A section not being present in the 10-K, the use of the wrong referencing format by the companies, or due to the algorithm. An example of wrong formatting is the case where "Item 7" and "Item 7A" for the MD&A section is being referenced to as "Items 7 and 7A"[5]. Some of these exceptions are repetitive and the code is adjusted accordingly. The RF section was not identified 4751 times of which most of these are for the year 2005 as the RF section was only obligatory from the 10-K filing in 2006 on for most companies. Extracting particular sections of a 10-K goes along with some trade-offs which are optimized by iterating over the 10-Ks and adjusting the extraction algorithm the best way possible.

Furthermore, I require the sections to have at least 250 words. One of the reasons for this is that several MD&A sections are already part of the annual report of a company and therefore 'incorporated by reference' as an exhibit filed alongside the 10-K. Identifying incorporated MD&A and RF sections for these 10-Ks is difficult as there is no structure at all in these referenced annual reports or exhibits.

2.3 Final 10-K sample

The final sample consists of 46,124 companies and 2072 matched takeovers for the MD&A section and of 43,948 companies and 1929 matched takeovers for the RF section, see Table 1. In Figure 3 the average number of words per section per year can be found (for sections with more than 250 words). It is clear that the length of the MD&A section stayed pretty much equal from 2008 until 2018, while the RF section size has been steadily growing since 2006 with on average more than 500 words per year. The reason for this is that the RF section is the section where all risks need to be described and companies want to disclose as few information as possible in their 10-Ks. Therefore, all risks, small or large, are written down in the Risk Factors section and finding which risks are actually informative is made hard by making the RF section larger annually.

[4]See `https://sraf.nd.edu/data/stage-one-10-x-parse-data/`

[5]See e.g. `https://www.sec.gov/Archives/edgar/data/1130464/000113046407000082/form10k_2006.htm`

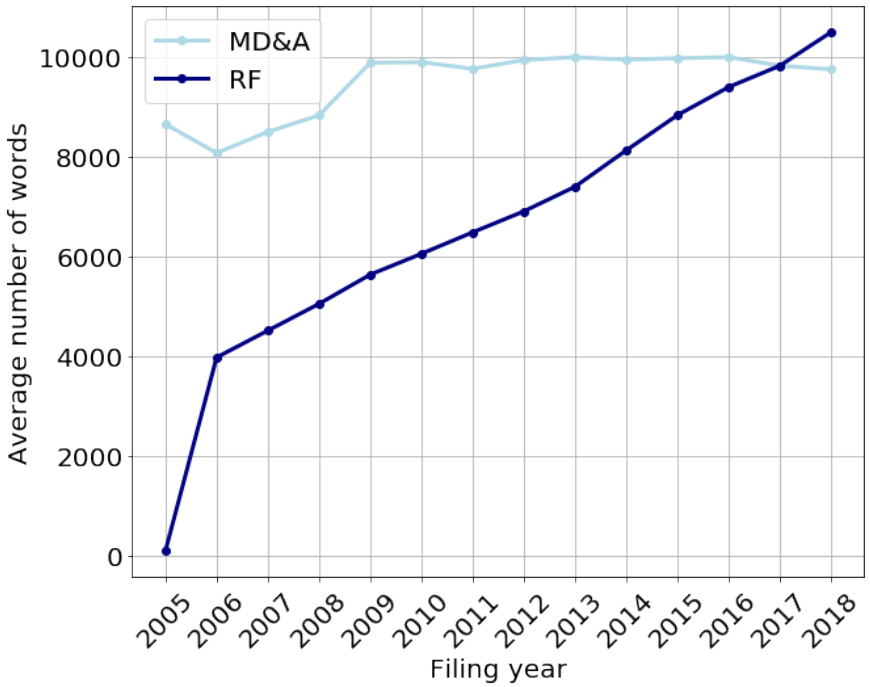

Figure 3: Average number of words in the MD&A and RF section per filing year.

2.4 Text preprocessing

In this part, the choices made regarding text preprocessing are discussed. In text analytics, text preprocessing is the task of preparing the textual data before encoding the texts. Text preprocessing removes words or numbers in textual data that do not give any additional meaning in the analysis. At this stage, all documents only contain actual alphabetical and numerical characters (words and numbers). In Section 2.2, the retrieval of both the RF and MD&A sections already uses some form of text preprocessing as all documents are in lowercase and punctuation is removed after extracting the textual information of the 10-Ks. The format of the extracted items from the 10-K is now a string of characters.

First, the title headers of the sections are removed using regular expressions. The reason for this is that, after retrieval, every RF and MD&A section still starts with "item 1a risk factors", "item 7 managements discussion and analysis" or something similar after extraction.

Next, all numbers are removed from the strings as, for example, the number '2007' does not give any meaning to a document other than a year. Furthermore, these numbers are usually quoted from tables that are also removed.

Then, the data only consists of actual words. In textual data, it is common to change words that have the same root word to the same word and to remove words that do not contain any document-specific information. This means all textual data is tokenized and lemmatized. Finally, stop words are removed. Tokenization is done by splitting the text in string format on whitespaces to create lists of words instead of a string of text. In NLP, one usually chooses to either lemmatize or stem the text. Lemmatization groups words that have the same meaning together depending on the context. Stemming follows an algorithm to reduce every word to its stem, which makes it less sophisticated but faster to compute. For example, the word *'saw'* in the sentences *'I have a saw'* and *'I saw you'* would be reduced to *'sa'* or *'saw'* depending on the stemming algorithm. However, using lemmatization the words would be changed to *'saw'* and *'see'* respectively. Therefore, I choose to lemmatize the words. All these list of words are lemmatized using the `NLTK` package available in Python, see Bird et al. (2009).

The last task is to remove stop words. Stop words are words that are common words in a language and exist without any explicit meaning. Examples of stop words are *'the'*, *'is'*, *'at'* and *'in'*. I use the financial stop words list, created by Loughran and McDonald (2011), which is specifically designed for 10-K stop words and I remove all these words from the MD&A and RF sections.

3 Methodology

In this section, I explain the document vectorization methods to vectorize the preprocessed MD&A and RF in the annual 10-K document and to classify these documents in the takeover or non-takeover target class. Predictions are created based on classification methods that need numerical input. Textual data is, obviously, not numerical and therefore textual documents need to be represented numerically first. Word and document embeddings are names for the collection of tools that are used in NLP to produce vectors from words or documents in a corpus of textual documents. A corpus is a set, or resource, of structured textual documents which can be used for statistical analyses. In my case, the corpus is the set of MD&A and RF documents. Using the vector representation of these documents, classification methods are used to analyze the predictive power of the models based on the percentage of identified takeover targets in the subset created, and investment portfolios are constructed and the returns of these portfolios are described with three models to study whether these portfolios outperform the market. The chosen document

vectorization methods, classification methods and factor models are described throughout this section in this particular order. Due to the small percentage (4.3% on average) of actual takeover targets per year in the sample, I expect the classification methods to classify almost all documents in the largest class, the non-takeover target class. Therefore, I choose to rank the probabilities or decision function outcomes of the document vectors of the companies per year and use create a portfolio using the companies with the highest rank.

3.1 Word and document vectorization

There are several methods to produce document vectors from textual data. Document vectorization methods are used to create document vectors for all documents in a corpus based on the words in the texts. In this section, I explain how three methods convert textual data to numerical vectors.

3.1.1 Tokenization

The preprocessed textual data of the MD&A and RF sections consists of a list of words for every document. These word lists are again tokenized using n-grams. n-grams are sequences of all n adjacent words in a document. For example, the sentence *'John likes walking with his dog.'* is lemmatized and preprocessed to *'john like walk with dog'*. Then, this preprocessed text is tokenized to a list of the following words: [*'john', 'like', 'walk', 'dog'*]. This tokenized list consists of the following list of unigrams (1-grams): [*'john', 'like', 'walk', 'dog'*], the following list of bigrams (2-grams): [*'john like', 'like walk', 'walk dog'*] and the following list of trigrams (3-grams): [*'john like walk', 'like walk dog'*], and so on.

3.1.2 Bag of Words model

The bag of words model (BoW) is a simple method to represent every document by a numerical vector by counting the frequency of tokens (term frequency) in every document by a so-called raw count.

Let P denote the number of documents in corpus D. Every document i, $i = 1, \ldots, P$, in corpus D is represented by a collection of tokens $\boldsymbol{d}_i = (x_i^1, \ldots, x_i^{p_i})$, where p_i is equal to the number of tokens in document i and x_i^l is defined as token l in document i. The vocabulary (of tokens) V is defined as the union of all unique tokens in the documents. This means $V = \bigcup_{i=1}^{P} \boldsymbol{d}_i$. Then the BoW model, or term frequency, is computed for every unique token $t \in V$ in the vocabulary for every

document i as

$$\text{tf}(\boldsymbol{d}_i, t) = \sum_{j=1}^{n_i} \mathbb{1}(t = x_i^j). \tag{1}$$

The vector representation of every document i is then equal to the number of occurrences of every token $t \in V$, where every unique token has the same entry for all documents.

In this thesis, I use BoW in combination with unigrams, bigrams and a combination of the two. Furthermore, I choose to use only the words that appear in more than 0.5% of the documents. The reason for that is that the vocabulary size of BoW largely affects the computation time of the classification algorithms.

3.1.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (tf-idf) reflects the importance of the appearance of a token or term in a document that is in a corpus of text documents. The reasoning behind tf-idf is that frequent tokens (words) that appear in a high percentage of documents contain less information about the document than words that appear less frequent in a small percentage of documents. Like the BoW model, it consists of the term frequency $\text{tf}(\boldsymbol{d}_i, c)$ which is then multiplied by the inverse document frequency $\text{idf}(t, D)$. The inverse document frequency $\text{idf}(t, D)$ is equal to the logarithmic inverse fraction of the number of documents token t appears in corpus D. For token t in document i in corpus D

$$\text{tf-idf}(\boldsymbol{d}_i, t, D) = \text{tf}(\boldsymbol{d}_i, t) \times \text{idf}(t, D), \tag{2}$$

where

$$\text{idf}(t, D) = \log \frac{P}{1 + |\{\boldsymbol{d}_i \in D : t \in \boldsymbol{d}_i\}|}, \tag{3}$$

where $|\{\boldsymbol{d}_i \in D : t \in \boldsymbol{d}_i\}|$ is the number of documents token t appears in and P the number of documents in corpus D. If a token does not appear in any document $|\{\boldsymbol{d}_i \in D : t \in \boldsymbol{d}_i\}| = 0$ and therefore it is common to add 1 to the denominator to prevent division by zero.

Just like with BoW, I use tf-id with unigrams, bigrams and a combination of the two. Furthermore, only the words that appear in more than 0.5% of all documents are used to reduce the vocabulary size and computation time.

3.1.4 Doc2Vec

The Doc2Vec algorithm is created by Le and Mikolov (2014). Doc2Vec is based on the Word2Vec algorithm developed by Mikolov et al. (2013). In contrast to BoW and tf-idf, both Word2Vec

and Doc2Vec use "context" words; a number of words both before and after the current word in a document. These "context" words are used to construct word vectors for the words in the document. Word2Vec uses a neural network with one hidden layer to create word vectors in a corpus of text documents. There are two different specifications of the Word2Vec algorithm: the Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model (SG), see the Appendix for full derivations and a visual representation of the architecture. CBOW predicts the current word by the surrounding words, the context. On the other hand, SG predicts the context, using the current word only. The algorithm computes vectors of length N for every word, where N is usually chosen between 100 and 1000. In this thesis, I use $N = 300$ as the vector size of all words and documents. Word vectors are mapped and updated in a $V \times N$ matrix W, where V is the size of the vocabulary. The produced word vectors are created so that words in the same context are close to each other in the multidimensional vector space, and there is a possibility to add vectors to get close to a word that is a combination of the words. To illustrate this, let's consider Word2Vec generated vectors for all words in the corpus. Then the words *'man', 'woman', 'king'* and *'queen'* are vectorized so that *'man'* and *'woman'*, and *'king'* and *'queen'* are close to each other in the vector space and that the combination of the word vectors *'king'* - *'man'* + *'woman'* will be closest to the word vector of *'queen'*. The word vectorization method Word2Vec was extended to Doc2Vec for document vectorization. Both the original paper regarding Doc2Vec by Le and Mikolov (2014) and Kim et al. (2019) find that the Doc2Vec document vectorization method outperforms document vectorization methods like BoW and tf-idf, which is the reason for using Doc2Vec next to BoW and tf-idf.

Word2Vec constructs only word vectors. Doc2Vec also constructs document vectors next to the word vectors. Doc2Vec uses a neural network with one hidden layer for creating both the word vectors and document vectors of chosen length N in a corpus of text documents. Word and document vectors are mapped and updated in a $V \times N$ matrix W and $P \times N$ matrix D respectively, where V is the size of the vocabulary and P is the total number of documents in the corpus. Like Word2Vec, the document vectors in Doc2Vec are created so that similar documents are close to each other in the vector space. The Doc2Vec model also comprises two models: Distributed Memory Model (DM) and Distributed Bag of Words Model (DBOW). Both CBOW and DM, and SG and DBOW are related. DM and DBOW are created to efficiently compute and update all document vectors next to the word vectors in the vocabulary as there are less documents in the corpus than words in the vocabulary. In the Appendix, the architecture and complete derivations of both

Word2Vec and Doc2Vec models are given.

Again, as with BoW and tf-idf, I only use words that appear in more than 0.5% of all documents to reduce the vocabulary size.

3.1.4.1 Doc2Vec: Distributed Memory Model

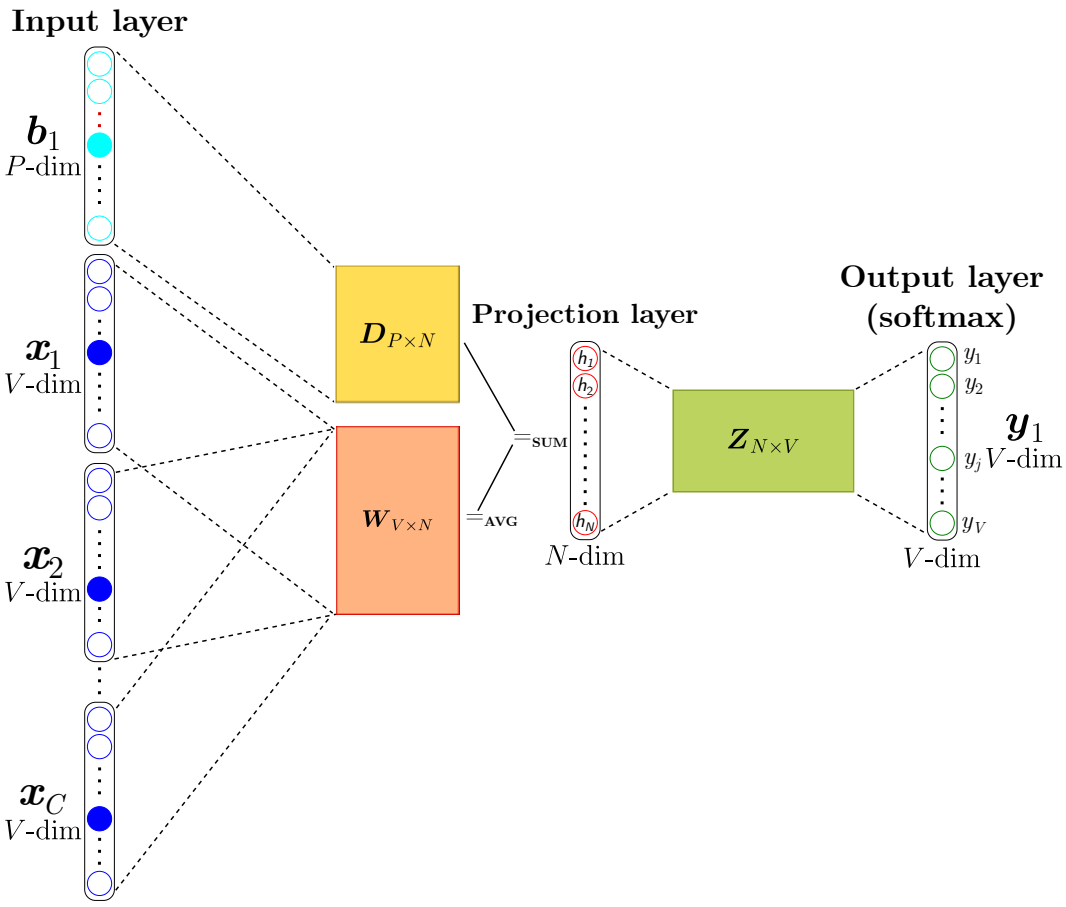

Figure 4: Architecture of the Distributed Memory Model.

The Doc2Vec based Distributed Memory Model (DM) is an extension of Continuous Bag of Words (CBOW) in Word2Vec. In this model, every document in a corpus is given as a one-hot encoded input vector $\boldsymbol{b}_1$ of size P, the number of documents in corpus D, in the input layer, next to the C context words. A one-hot encoded vector is a vector of zeros with a 1 at one of the elements. Therefore, $\boldsymbol{b}_1$ is a P-dimensional vector with a 1 at the q-th element for the q-th document in the corpus. The document vectors are updated by back-propagation in the $P \times N$ matrix $\boldsymbol{D}$, where each row represents a document vector. Next to that, the one-hot encoded input vectors of the C context words are given in the input layer by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_C$, and the word that is surrounded by those

words is used in the output layer as the one-hot encoded vector $\boldsymbol{y}_1$. All words in all documents are used for updating the word vector matrices in $\boldsymbol{W}$ and $\boldsymbol{Z}$, and document vector matrix $\boldsymbol{D}$. Figure 4 shows the architecture. The model is trained using the input layer for all words from all documents in the vocabulary. For all words in the vocabulary, the number of epochs (iterations) is chosen as 10, which is common in the literature.

The word vectors are derived from $\boldsymbol{W}$ and $\boldsymbol{Z}$. Therefore, the updates of matrix $\boldsymbol{W}$ and $\boldsymbol{Z}$ are important. These updates are equal to the updates in the CBOW model and full derivations can be found in the Appendix.

However, the document vectors are given in matrix $\boldsymbol{D}$. Therefore, the update for matrix $\boldsymbol{D}$ is important. The full derivation of the update is given in the Appendix, just like all other derivations of the underlying two-layer neural network of the Distributed Memory Model.

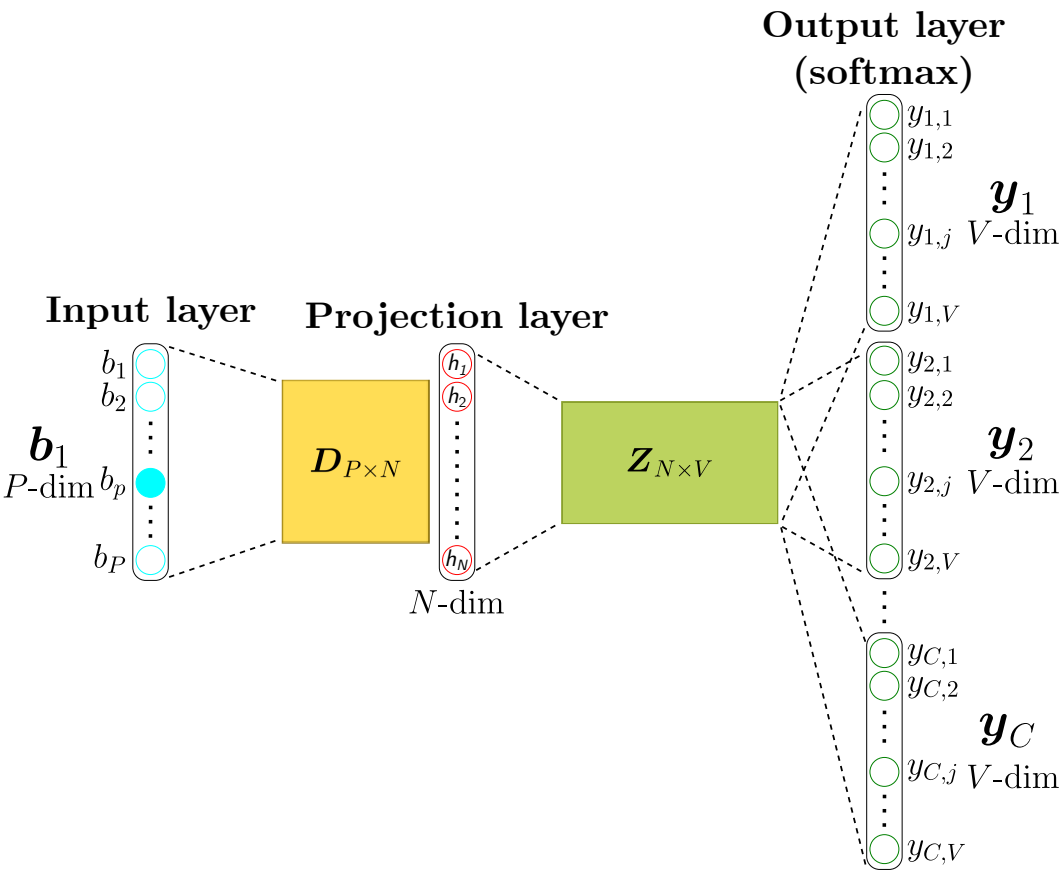

Figure 5: Architecture of the Distributed Bag of Words Model.

3.1.4.2 Doc2Vec: Distributed Bag of Words Model

Distributed Bag of Words Model (DBOW) is analogous to Skip-gram (SG) in Word2Vec. Instead of updating the word vectors matrix, or word embeddings, in matrix $\boldsymbol{W}$ (see Appendix) for all words in the vocabulary given the C surrounding words, DBOW updates the $P \times N$ document matrix $\boldsymbol{D}$ which makes DBOW more computationally efficient than SG and DM. The input is now only the one-hot encoded document vector $\boldsymbol{b}_1$, which is estimated using C randomly selected context words, represented by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_C$, that appear in the document that is represented by $\boldsymbol{b}_1$ in Figure 5. The random selection of the words in DBOW is the main difference between SG, as the input layer consists of the one-hot encoded word vector $\boldsymbol{y}_1$, which is estimated by C context words represented in the output layer by a one-hot encoded vector.

In Figure 5, the full architecture of DBOW is presented visually. Again, the number of epochs is 10 and the full derivation of the back-propagation for the underlying two-layer neural network of DBOW can be found in the Appendix.

3.1.4.3 Hierarchical softmax

The softmax functions in DM and DBOW compute probabilities for all V words in the vocabulary, which is equal to computational complexity $O(V)$. To decrease this complexity, two different approximations described in Morin and Bengio (2005) are used: the hierarchical softmax and negative sampling. The complexity $O(V)$ can be decreased to $O(\log_2 V)$ with hierarchical softmax which drastically increases computational efficiency. Morin and Bengio (2005) were the first to introduce the hierarchical softmax method in the neural network context.

In the hierarchical softmax, described in Mikolov et al. (2013), the output layer words are represented in a binary Huffman tree as the leaves. The child nodes of every node are then represented as relative probabilities, which assign probabilities to all the words using a random walk. From the root of the tree, one can find each word w by following a certain path. The path from the root of the binary tree to word w has $L(w)$ nodes and $n(w, i)$ is the i-th node on this path. This means that $n(w, 1) =$ root and $n(w, L(w)) = w$. Let $ch(n)$ be defined as the left child of node n. The probability of word w begins equal to the output word w_O

$$p(w = w_O) = \prod_{i=1}^{L(w)-1} \sigma([\![n(w, i+1) = ch(n(w, i))]\!] \cdot \boldsymbol{u}_{n(w,j)}^T \boldsymbol{h}), \tag{4}$$

where $[x]$ equals 1 if x is true and equals -1 else, $\boldsymbol{u}_{n(w,j)}$ is the vector representation of the word

at node $n(w, j)$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$. As $\sum_{k=1}^{V} p(w_k = w_O) = 1$, it can be verified that the cost of computing $\log p(w_O|w_I)$ and its gradient are proportional to $L(w_O)$ and this results in the computational complexity to decrease to less than $\log(V)$ on average.

3.1.4.4 Negative sampling

Negative sampling is another approximation of the softmax output layer. The negative samples are in the Doc2Vec cases words that are in fact wrong (negative) words for that position. Negative sampling is a simplification of Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010) that still contains the information needed to retain the quality of the word embeddings. Negative sampling replaces $\log p(w_O|w_I)$ with

$$\log \sigma(\boldsymbol{u}_{w_O}^T \boldsymbol{t}_{d_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \Big[\log \sigma(-\boldsymbol{u}_{w_i}^T \boldsymbol{t}_{d_I}) \Big] \tag{5}$$

in DBOW, where k is the number of negative samples, i.e. words that are not the actual output word, drawn using logistic regression from the noise distribution $P_n(w)$. Mikolov et al. (2013) indicate that smaller datasets require k between 5 and 20 and larger datasets only require k between 2 and 5, therefore I choose $k = 5$.

3.2 Classification methods

Based on the vector representations of the documents, I classify a train and test dataset to create different investment strategies based on the highest probability of being a takeover target. In my analysis, I have only two classes: the takeover targets labeled as "1" and the non-takeover targets labeled as "0" and the variables are defined by the document vectors produced by the three vectorization methods. I choose to predict the takeover targets for a certain year based on training on the 5 years prior to that year. In Figure 6 this setup is visualized. The portfolio is constructed based on the identified takeover targets per year.

I choose to use three different classification methods, the classic logistic regression classification, the multinomial naive Bayes in combination with only BoW and tf-idf as all variables should be higher than 0 for MNB, and support vector machines with RBF kernel. These three classification methods are widely used in the text classification literature. For all classification methods, I only use the document vectors as explanatory variables. Next to that, I use all the features of the document vectors for the classification and prediction as feature selection is time-consuming with a large vocabulary.

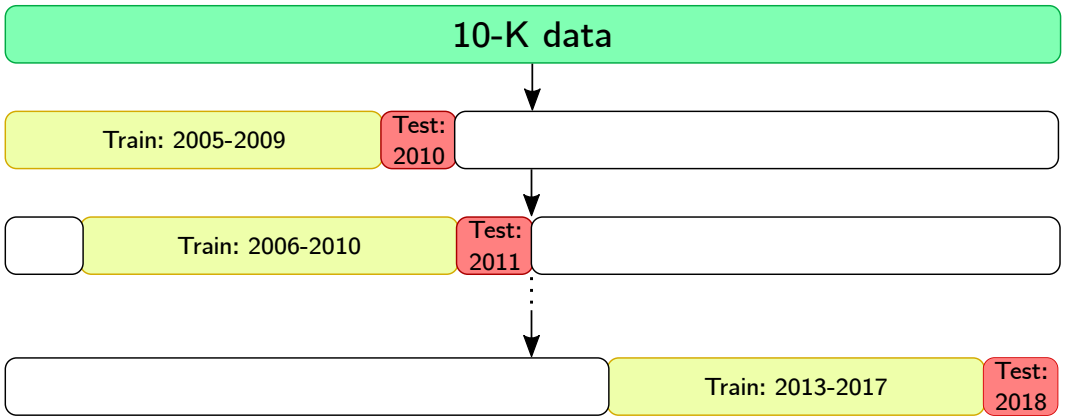

Figure 6: Train and test subsets for identifying takeover targets based on textual content in 10-Ks annually.

3.2.1 Logistic regression

Logistic regression is a classification method that uses a binary vector of the dependent variable $\boldsymbol{y}$ and the independent variables $p \times n$ matrix $\boldsymbol{x}$ consisting of the train data-points $\{y_i, \boldsymbol{x}_i\}_{i=1}^{n}$. The difference between logistic and linear regression is that in logistic regression the outcomes are weighed to probabilities between 0 and 1 using the sigmoid function

$$S(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}. \tag{6}$$

In my case, the independent variables $\boldsymbol{x}$ of the training set are equal to the document vectors created by the different document embedding models of all documents in the five years previous to the year predicted. For example, every document vector in the oW model is equal to a V-dimensional vector with V the vocabulary size. Then, the dependent variable $\boldsymbol{y}$ is the binary value for the company belonging to each document vector being a takeover target in the 12 months after the end of the month of the filing date of the 10-K. The model is fitted on the train data-points and used for estimation using test data-points $\{\boldsymbol{x}_j\}_{j=1}^{m}$, which estimates the probability $0 \leq y_j \leq 1$ for all $j = 1, \ldots, m$ of belonging to class 1: the takeover targets.

The linear regression model fits the train data-points $\{y_i, \boldsymbol{x}_i\}_{i=1}^{n}$ to

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i \text{ for } i = 1, \ldots, n. \tag{7}$$

Then the logistic regression model weighs the linear regression model using the sigmoid function to

$$P(y_i = 1) = \frac{1}{1+\exp(-(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}_1))} = \frac{\exp(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}_1)}{1+\exp(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}_1)} \text{ for } i = 1, \ldots, n, \tag{8}$$

where $P(y_i = 1)$ is the probability of y_i belonging to class 1, the takeover targets. From this, the log-odds of the event $y_i = 1$ is derived by

$$l = \ln \frac{P(y_i = 1)}{1 - P(y_i = 1)} = \ln \frac{P(y_i = 1)}{P(y_i = 0)} = \beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}_1 \text{ for } i = 1, \ldots, n. \tag{9}$$

The model parameters β_0 and $\boldsymbol{\beta}_1$ are estimated by maximizing the log-likelihood function

$$l(\beta_0, \boldsymbol{\beta}_1) = \sum_{i=1}^{n} y_i \ln(P(y_i = 1)) + (1 - y_i) \ln(1 - P(y_i = 1)), \tag{10}$$

with respect to all individual parameters and where $P(y_i = 1)$ is known for all n training data-points as either $P(y_i = 1) = 0$ or $P(y_i = 1) = 1$.

Then, for all test data-points $\{\boldsymbol{x}_j\}_{j=1}^m$ the probability that the data point is equal to 1 is estimated by

$$P(y_j = 1) = \frac{1}{1 + \exp(-(\beta_0 + \boldsymbol{x}_j^T \boldsymbol{\beta}_1))} = \frac{\exp(\beta_0 + \boldsymbol{x}_j^T \boldsymbol{\beta}_1)}{1 + \exp(\beta_0 + \boldsymbol{x}_j^T \boldsymbol{\beta}_1)} \text{ for } j = 1, \ldots, m. \tag{11}$$

3.2.2 Multinomial Naive Bayes

Naive Bayes classifiers are a set of classification algorithms based on Bayes' theorem. Naive Bayes uses the assumption that all features in the data are independent. This independence assumption is the reason for naming this method 'naive' as this assumption is usually not true. However, this assumption is useful and usually a good approximation of reality. For class $c \in C$ and the document vector $\boldsymbol{d} = (d_1, \ldots, d_V)$ with V the size of the vocabulary Bayes' theorem states that

$$P(c|\boldsymbol{d}) = \frac{P(c)|P(\boldsymbol{d}|c)}{P(\boldsymbol{d})} = \frac{P(c, \boldsymbol{d})}{P(\boldsymbol{d})}. \tag{12}$$

In our case, $C = \{0, 1\}$ and vector $\boldsymbol{d}$ is equal to a document vector of length V created by BoW or tf-idf. Only for BoW and tf-idf probabilities of events are vectorized as BoW simple counts words and tf-idf counts words and divides it by the document frequency. As the Doc2Vec algorithms do not generate document vectors with probabilities of events (words in this case), but document specific vectors of chosen length $N = 300$, the MNB classifier is only used in combination with BoW and tf-idf document vectorization methods.

The denominator $P(\boldsymbol{d})$ is independent of c and $\boldsymbol{d}$ is given, which results in

$$P(c|\boldsymbol{d}) \propto P(c)|P(\boldsymbol{d}|c) = P(c, \boldsymbol{d}) \tag{13}$$

as $P(\boldsymbol{d})$ is constant in (12). Then, using the definition of conditional probability in combination with the 'naive' assumption that all words $d_1, \ldots, d_m$ in the document vector are independent conditional on c the numerator of (12) can be written as

$$\begin{aligned} P(c, \boldsymbol{d}) &= P(d_1, \ldots, d_V, c) \\ &= P(d_1|d_2, \ldots, d_V, c)P(d_2, \ldots, d_V, c) \\ &= P(d_1|c)P(d_2|d_3, \ldots, d_V, c)P(d_3, \ldots, d_V, c) \\ &= \ldots \\ &= P(d_1|c)P(d_2|c)\ldots P(d_{V-1}|c)P(d_V|c)P(c) \\ &= P(d_1|c)P(d_2|c)\ldots P(d_{V-1}|c)P(d_V|c)P(c) \\ &= P(c)\prod_{i=1}^{V} P(d_i|c), \end{aligned} \tag{14}$$

where $\prod_{i=1}^{V} P(d_i|c)$ can simply be observed in the data by counting Substituting in (13) results in the General Naive Bayes

$$P(c|\boldsymbol{d}) \propto P(c)\prod_{i=1}^{V} P(d_i|c). \tag{15}$$

The multinomial naive Bayes (MNB) classifier is a specific case of Naive Bayes. MNB uses a multinomial distribution for all features, in our case words. The probability of obtaining document $\boldsymbol{d}$ in class c is given by

$$P(\boldsymbol{d}|c) = \frac{\sum_{i=1}^{V} d_i!}{\prod_{i=1}^{V} d_i!}\prod_{i=1}^{V} p_i^{d_i}, \tag{16}$$

where p_i is equal to the proportion of documents that word i appears in given class c. Then, 13) results in

$$P(c|\boldsymbol{d}) \propto P(c)\frac{\sum_{i=1}^{V} d_i!}{\prod_{i=1}^{V} d_i!}\prod_{i=1}^{V} p_i^{x_i} \tag{17}$$

for the MNB classifier.

3.2.3 Support Vector Machines

Support Vector Machines (SVMs) are a set of supervised classification models created by Vapnik. An SVM fits the training data-points to a hyperplane, which allows for classifying test data-points to one of the two classes.

Assume there are n P-dimensional training vectors such that $\boldsymbol{x}_i \in \mathbb{R}^p$ and the dependent variable $y_i \in \{-1, 1\}$ for $i = 1, \ldots, n$. Cortes and Vapnik (1995) define the following optimization problem

$$\begin{aligned} \min_{w,b,\psi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\psi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \psi_i \\ & \psi_i \geq 0 \text{ for } i = 1, \ldots, n, \end{aligned} \tag{18}$$

where $\boldsymbol{w} \in \mathbb{R}^p$ and $b \in \mathbb{R}$, $\phi(\boldsymbol{x}_i)$ is a mapping function that can be either linear or nonlinear, and penalty term $C > 0$. The hyperplane that divides the data is defined as $\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b$. (18) is called the primal problem. The penalty term C is a trade-off between smooth decision boundary (small $C < 1$) and classifying the training points correctly (large $C > 1$). The parameter therefore needs to be tuned for the best performance. Parameter C is thus called a hyperparameter and selected for every individual model based on certain properties explained in Section 4.1. The goal of SVM is to find the best separation between classes and ideally $y_i(\boldsymbol{w}^T\sigma(\boldsymbol{x}_i + b) \geq 1$ would hold for all training data-points, which means every training data-point is on the correct side of the decision boundary. Though, the hyperplane, or decision boundary, is usually not able to perfectly separate and predict the training data-points. Therefore, some samples are allowed from the margin boundary with deviation ψ_i. To efficiently solve the primal problem, it is rewritten by solving for the Lagrangian dual to obtain the simplified dual problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq C \text{ for } i = 1, \ldots, n, \end{aligned} \tag{19}$$

where $\boldsymbol{e}$ is an n-dimensional vector of ones, the $n \times n$ matrix $\boldsymbol{Q}$ is positive semidefinite. Originally, $\phi(\boldsymbol{x}_i)$ was defined as $\boldsymbol{x}_i$ which resulted in the classical linear SVM. Though, the kernel trick defines the elements of matrix $\boldsymbol{Q}$ by $Q_{ij} \equiv y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$. Kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is used to create a high dimensional space by mapping the training vectors and can be defined by different functions. The original linear SVM is defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$. In this study, I use the radial basis function (RBF)/Gaussian kernel defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\big(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\big) = \exp(-\gamma \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2)$, where $\gamma = \frac{1}{2\sigma^2}$ is another parameter, next to C, that needs to be tuned.

After optimizing the dual problem and finding the hyperplane that divides the data best, the

prediction of class y for test vector $\boldsymbol{x}$ is computed by

$$\text{sign}(\sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \geq 0 \\ \text{-1} & \text{if } \sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b < 0, \end{cases} \tag{20}$$

Instead of predicting the actual class (1 or -1) for every test data-point, only decision function

$$\sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \tag{21}$$

is used to decide which companies have the highest chance of being a takeover target based on the textual data in the MD&A or RF section. The highest (lowest) obtained value from the decision function has the highest probability of belonging to class 1 (0, or -1 in SVM).

3.3 Models

In Figure 7, the diagram of models is visually displayed. I study two different sections of the 10-Ks: the MD&A and RF section. Both sections are used in combination with BoW, tf-idf, or one of the two Doc2Vec algorithms; DBOW and DM.

The BoW and tf-idf document vectorization methods are combined with unigrams (1-grams), bigrams (2-grams) and the combination of all unigrams and bigrams (1- and 2-grams). This results in 6 different specifications: BoW with unigrams (BoW-1), BoW with unigrams and bigrams (BoW-12), BoW with bigrams (BoW-2), tf-idf with unigrams (tf-idf-1), tf-idf with unigrams and bigrams (tf-idf-12), and tf-idf with bigrams tf-idf-2.

The two Doc2Vec methods are used in combination with one of the two softmax approximations: hierarchical softmax (HS) and negative sampling (NS). This results in 4 different specifications: DBOW with hierarchical softmax (DBOW-HS), DBOW with negative sampling (DBOW-NS), DM with hierarchical softmax (DM-HS), and DM with negative sampling (DM-NS).

The six document vectorization methods based on BoW and tf-idf are used with the three different classification methods: logistic regression, multinomial naive Bayes and SVM with RBF kernel. This results in a total of 18 models for both the MD&A and RF section.

The four different Doc2Vec models specifications are combined with two classification methods: logistic regression and SVM with RBF kernel. As explained in Section 3.2.2, multinomial naive Bayes is based on probabilities of events and is therefore only used in combination with BoW and tf-idf as Doc2Vec does not vectorize documents based on probabilities of events. This results in a total of 8 models for both the MD&A and RF section, which adds up to 52 different models created by combining the two sections, three document vectorization methods and three classification methods.

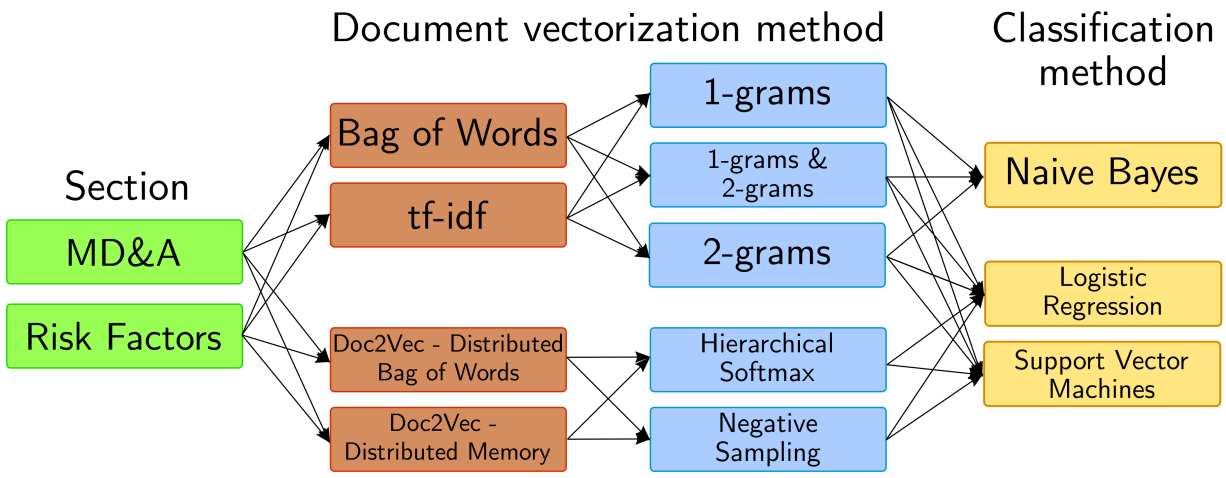

Figure 7: All combinations of the section with document vectorization methods and classification methods. All lines represent a combination. In total, 52 models are used.

3.4 Top Decile Function

The classification methods used are estimating probabilities or use the decision function for the SVM to rank the most likely takeover targets for all companies in a given year based on the training set of the 5 years previous. As mentioned, the portfolios are created by ranking the probabilities or decision function of the classification methods. The top decile function is used to show how well the models predict in a subset with the highest ranked documents in terms of probability or decision function. The top decile function divides a set of probabilities or decision function values in 10 parts by ranking the values and selects the 10% highest values.

Assume that for year $y = 2010, \ldots, 2018$ there are n_y companies in the total test set. Then, the companies are ordered by predicted probability for every classification method m in year y so that the companies are described by the set: $\{p_{y,m,i}, i = 1, \ldots, n_y : p_{y,m,i+1} \geq p_{y,m,i}\}$, where $p_{y,m,1}$ is the company with the lowest and p_{y,m,n_y} the company with the highest probability for classification method m in year y. Then, the top decile function selects the companies using the subset of companies belonging to the highest $\lfloor \frac{10}{100} \times n_y \rceil$ probabilities, where $\lfloor x \rceil$ rounds x to the nearest integer. This equates to selecting company $p_{y,m,i}$, where $i = n_y - \lfloor \frac{10}{100} \times n_y \rceil$. This means for every classification method the top 10 percent of highest predicted probabilities or highest decision function values of documents belonging to the takeover targets are added to the portfolio per year. Table 3 shows the number of companies selected every year based on the top decile

function. All portfolios consist of 2945 and 3001 companies for portfolios for the MD&A and RF section respectively. Next to that, Table 3 shows the actual number of takeover targets in the full sample alongside the percentage of actual takeover targets in the data per year.

The portfolio that is created using the top decile function is invested in using an equally-weighted portfolio. This means every company selected using the top decile function has the same amount of investment, for example 1 dollar. Actually investing in stocks, means I hold a long position for all companies selected for the duration of exactly 12 months starting at the end of the month of the filing date of the 10-K of a company. This means all 2945 and 3001 companies in the portfolio of the MD&A and RF section models are in the portfolio for exactly 12 months with the same amount of investment, and based on this the portfolios are analyzed.

The portfolio is used to calculate the percentage of identified takeover targets and to estimate the performance on the markets based on the excess returns.

Table 3: Number of companies per year, the number of companies in the portfolio per year, the number of takeovers per year and the takeover percentage per year for the for both the MD&A and RF section samples.

	MD&A				Risk Factors			
Year	N	Top Decile N	Takeovers	%	N	Top Decile N	Takeovers	%
2010	3173	317	161	5.1	3278	328	161	4.9
2011	3191	319	126	3.9	3284	328	127	3.9
2012	3114	311	140	4.5	3198	320	137	4.3
2013	3150	315	126	4.0	3217	322	131	4.1
2014	3355	336	140	3.9	3403	340	135	4.0
2015	3453	345	166	4.8	3498	350	167	4.8
2016	3363	336	171	5.1	3398	340	170	5.0
2017	3310	331	127	3.8	3353	335	130	3.9
2018	3353	335	124	3.7	3375	338	128	3.8
Total	**29462**	**2945**	**1271**	**4.3**	**30004**	**3001**	**1286**	**4.3**

3.5 Factor models

To measure the performance of a portfolio, the excess return, called alpha (α), of the portfolio relative to a market index is measured, alongside the volatility or risk, called beta (β). There are several different models to measure alpha, also known as factor models. In this study, I use the Capital Asset Pricing Model (CAPM) to derive Jensen's alpha (α^J) developed in the 1960's by, among others, Sharpe (1964) and Lintner (1965), the Fama-French Three-Factor Model (FF3FM) and the Carhart Four-Factor Model (C4FM), which both extend CAPM using additional factors which help to add to the explanatory power of Jensen's alpha.

3.5.1 Capital Asset Pricing Model

CAPM is an investment theory that identifies the relationship between the systematic market risk and the expected return of an asset or portfolio. The theory is based on the fact that risk always exists in a portfolio, no matter how diversified the portfolio is. Therefore, the expected return of asset or portfolio $E(R_i)$ is based on the risk an investor makes with the asset or portfolio. Mathematically, CAPM can be written by

$$E(R_i) = R_{rf} + \beta_i(E(R_m) - R_{rf}) = R_{rf} + \beta_i MRP, \tag{22}$$

where R_{rf} is the risk-free rate and $E(R_m)$ is the expected return of the market, which is equal to a market index like the large-cap benchmark S&P 500. The only factor in this model is the market risk premium $E(R_m) - R_{rf} = MRP$, which is the expected return of the market minus the risk-free rate, based on the Treasury bill-rate, and a measure of the return demanded for the risk for investing. β_i describes the volatility or market risk of asset or portfolio i, where $\beta_i = 1$ represents equal volatility of the investment portfolio i as opposed to the benchmark index m, $\beta_i = 1.5(0.5)$ represents a volatility of 150% (50%) compared to the chosen index which makes it more (less) risky, and negative β_i means the investment portfolio has an inverse relation with the market index. Though, CAPM explains only 70% of the return of the portfolio and therefore other models with more explanatory power are preferred.

3.5.2 Jensen's alpha

Shortly after the introduction of CAPM, Jensen (1968) extended CAPM by simply adding an intercept to CAPM which is used to evaluate the performance of a mutual fund. This resulted in

$$R_i = \alpha^J + R_{rf} + \beta_i(R_m - R_{rf}) = \alpha^J + R_{rf} + \beta_i MRP, \tag{23}$$

where R_i is the actual return of an asset and α^J the intercept added which measures systematic risk-adjusted returns. The market risk premium MRP is, just like the asset return R_i, calculated based on actual performance of a benchmark index. As CAPM and Jensen's alpha are both risk-adjusted, significant $\alpha^J > 0 (< 0)$ results in a significant higher (lower) asset return than the risk adjusted return, which means the asset outperforms (performs worse than) the market. For example, if an investment portfolio is created using monthly stock prices and monthly market risk premium prices, if α^J is significantly larger than 0, for example, 0.50, then the portfolio would outperform the benchmark with 50 basis points per month, which results in 617 basis points or 6.17%[6] per year, where a basis point is equal to 0.01%.

An investment portfolio consists of multiple assets and uses a regression on all data points to estimate α^J and β_i.

3.5.3 Fama-French Three-Factor Model

For a long time, Jensen's alpha was used to describe excess portfolio returns (α) and systematic market risk(β_i). In 1992, the Fama-French Three-Factor Model (FF3FM) was developed by Fama and French (1992) who added two factors to CAPM, Small Minus Big (SMB) which measures small over big capitalization historic excess returns and High Minus Low (HML) which measures value stocks over growth stocks, and includes, like Jensen, the intercept α^{3F} which describes the excess return. The reason for the introduction of the FF3FM is the fact the explanatory power of the model jumped from around 70% to more than 90% for a portfolio's return. The time-series regression of the FF3FM can be written as

$$R_{i,t} = \alpha^{3F} + R_{rf,t} + \beta_1 MRP_t + \beta_2 SMB_t + \beta_3 HML_t + \varepsilon_t, \tag{24}$$

where R_{it} is the realized return of stock or portfolio i at time t, MRP_t the market risk premium of the benchmark index at time t, SMB_t the SMB value at time t and HML_t the HML value at

[6]For annualizing the monthly excess return the following formula is used: $((1 + R/100)^{12} - 1) * 100$, where R represents the monthly excess return

time t[7]. The excess return is explained by α^{3F} and if $\alpha^{3F} > 0 (< 0)$ is significant, the investment portfolio beats (performs worse than) the benchmark index with the assumed risk β_1. If $\alpha^{3F} = 0$ the investment portfolio has a normal return for the risk taken compared with the benchmark portfolio. If the time series regression in (24) is for a monthly portfolio, α^{3F} represents the excess return in percentage on a monthly basis.

3.5.4 Carhart Four-Factor Model

Carhart (1997) introduced an extension to the FF3FM named the Carhart Four-Factor Model (C4FM) with another factor: the monthly momentum factor (MOM), sometimes referred to as Up Minus Down (UMD), which measures winners over the past 12-months minus losers over the past 12-months. The time-series regression of C4FM can be written as

$$R_{i,t} = \alpha^{4F} + R_{rf,t} + \beta_1 MRP_t + \beta_2 SMB_t + \beta_3 HML_t + \beta_4 MOM_t \varepsilon_t, \tag{25}$$

where α^{4F} describes the excess return in C4FM, MOM_t[8] the MOM value at time t and the rest of the variables are defined as in (24). The excess return is explained by α^{4F} and if $\alpha^{4F} > 0 (< 0)$ is significant, the investment portfolio beats (performs worse than) the benchmark index with the assumed risk β_1. If $\alpha^{4F} = 0$ the investment portfolio has a normal return for the risk taken compared with the benchmark portfolio.

3.5.5 Benchmark index

All three models (Jensen's alpha, FF3FM and C4FM) require the variable MRP_t to be the market risk premium of a benchmark index. In Kenneth French's data library the variable $Mkt - RF$ can be found, which is the value-weighted return on all NYSE, AMEX and Nasdaq stocks minus the one-month Treasury bill rate, the risk-free rate. However, I use the Russell 3000 index as penny stock companies and companies with a very low market capitalization are removed from the sample, see Table 1. The Russell 3000 index tracks the performance of the 3000 largest, in terms of market capitalization, traded stocks in the United States. The Russell 3000 index includes around 98% of all equity securities and is value-weighted, which means the higher the total market value of the company, the higher the proportion in the index.

[7]The time-series data for the risk-free rate R_{rft}, SMB and HML can be found in Kenneth French's data library: `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`.

[8]The time-series data for the monthly momentum (MOM) can also be found in Kenneth French's data library.

4 Results

In this section, the results of the different classification methods are described. First, parameter C and γ are tuned for the SVMs. Then, portfolios are constructed based on the document vectors and classification methods. The classification methods identify the most likely companies that are a takeover target per year and construct portfolios using the top decile function based on the most likely takeover targets from 2010 to 2018. The portfolios are analyzed based on their prediction performance of takeover targets in the portfolio and the portfolio performance in the markets.

4.1 Hyperparameter tuning

As described, parameter C and γ is tuned for SVM with the RBF kernel. In this study, I use three different document vectorization methods, BoW, tf-idf and Doc2Vec. A higher C, for γ constant, results in the SVM choosing a smaller margin for the hyperplane to fit the training data-points, and vice versa.

Due to the top decile function, parameter C and γ should also be chosen based on the prediction performance of the top 10 percent most likely documents to be a takeover target in the test data. Fitting the hyperplane of the RBF kernel in an SVM is time-consuming and the original cross-validation for hyperparameterization is not manageable. Therefore, I choose to iterate 10 times on a small subset of 5000 training data-points to fit the SVM. The decision function of the SVM is used to create probabilities for the 1000 test data-points. Then, the top decile function is used to obtain the 10% highest ranked test data-points. These are the 10% most likely takeover targets in the subset. This results in 100 companies to be selected for all 10 iterations from the test data. The default value for $C = 1$ and for $\gamma = \frac{1}{\#\text{ of variables}}$, where the number of variables is equal to the total number of unigrams, bigrams and combination of unigrams and bigrams which appear in more than 0.5% of the documents for BoW-1 and tf-idf-1, BoW-2 and tf-idf-2, and BoW-12 and tf-idf-12, respectively, and equal to the number of words appearing in more than 0.5% of the documents for the Doc2Vec methods. C and γ are chosen based on the highest average percentage of actual takeover targets in the 10 subsets created using the top decile function on the 1000 test data-points.

Analyses shows that the decision function of the SVM creates equal decision function values for all test data-points if $C > 1$ or $\gamma > 1$ when using the BoW and tf-idf document vectorization methods. This is probably due to the fact that every document vector is high-dimensional and that

the document vectors are quite similar in BoW and tf-idf, which classifies all documents to the non-takeover target class more strictly. Therefore, for the models including BoW and tf-idf the value of C is chosen from 10^{-3} to 1 (10^0), and for γ from 10^{-6} to 1 (10^0). It is standard procedure to choose different values for both C and γ as powers of 10. This results in a grid of 28 combinations of C and γ for which the average identified percentage of takeover targets is calculated for 10 iterations. The combination which gives the highest takeover target percentage in the 10 iterations is used in the actual model for the document classification.

The models with Doc2Vec document vectorization methods create 300-dimensional vectors, which is small compared to the V-dimensional document vectors created by BoW and tf-idf . The problem that occurs for $C > 1$ for all methods including BoW and tf-idf does not appear in the models including Doc2Vec. Therefore, the value of C is chosen from 10^{-3} to 10^3, and the value of γ

Table 4: Chosen hyperparameters for C and γ, and the average percentage of takeover targets in the portfolio of 100 companies selected using the top decile function on the 1000 test data-points for 10 iterations for all SVMs for all document vectorization methods based on the MD&A and Risk Factors sections. C and γ are chosen based on the highest average percentage of takeover targets in the 10 portfolios created in 10 iterations using the top decile function on the 1000 test data-points.

	MD&A			Risk Factors		
Vectorization	Portfolio %	C	γ	Portfolio %	C	γ
BoW-1	6.2	0.001	10^{-6}	5.4	0.001	1
BoW-12	6.1	0.1	10^{-4}	5.6	1	0.01
BoW-2	7.0	0.1	1	6.3	0.01	1
tf-idf-1	5.5	0.01	0.1	5.3	1	0.001
tf-idf-12	7.1	0.1	1	6.0	1	0.01
tf-idf-2	8.1	1	1	6.7	1	1
Doc2Vec - DM-HS	6.8	100	10	7.3	10	10
Doc2Vec - DM-NS	7.3	100	10	7.3	100	0.001
Doc2Vec - DBOW-HS	6.6	10	10	7.3	10	10
Doc2Vec - DBOW-NS	7.4	0.001	0.1	6.9	0.01	0.01

from 10^{-6} to 10^{2}. This results in a grid of 56 combinations of which the highest average percentage is chosen for the actual document classification.

In Table 4, the selected C and γ, and the average percentage of identified takeover targets in the 10 iterations are shown for all document vectorization methods based on the MD&A and Risk Factors sections which are selected based on the highest average percentage of actual takeover targets among the 100 selected companies in 10 iterations. The document vectorization methods are both BoW or tf-idf with unigrams (1-grams) and/or bigrams (2-grams), or Doc2Vec which includes Distributed Memory and Distributed Bag of Words using either hierarchical softmax or negative sampling.

Table 4 shows that the tf-idf document vectorization method with bigrams based on the MD&A section has the highest average takeover percentage in the 10 iterations of 8.1%. This means that, on average, 8.1% of the companies selected in the portfolio based on the top 10% most likely takeover targets are actually a takeover target. The percentage of takeover targets in the total samples of both the MD&A and RF section is equal to 4.3%. Next to that, BoW and tf-idf document vectorization methods with unigrams are outperformed by the combination of unigrams and bigrams, except for BoW-1 and BoW-12, where the average percentage of BoW-1 is 6.2% and BoW-12 is 6.1%. Subsequently, BoW and tf-idf combined with the combination of unigrams and bigrams is outperformed by the models based on bigrams only. Furthermore, all BoW and tf-idf document vectorization methods have a higher average percentage of takeover targets for the MD&A based models compared to the Risk Factors based models.

The Doc2Vec models based on the Risk Factors section have the highest average takeover percentage of all document vectorization methods based on the Risk Factors section with an average of 6.9% of actual takeover targets in the portfolios created for DBOW-NS and 7.3% of actual takeover targets in the portfolios created for DM-HS, DM-NS and DBOW-HS.

4.2 Takeover percentage and portfolio performance

There are two indications of a good performance of the subset created by the different models. Table 5 shows the percentage of takeover targets in the portfolios (portfolio %) and the percentage of identified takeover targets of all takeover targets in the data that are selected in the portfolio (identified %), which are the first indicator. The annualized values for the portfolio returns α^{3F} and α^{4F} are also shown in Table 5, which is the second indication. The higher the percentage of takeover targets in the portfolio or actual identified takeover targets, the better the model identifies

takeovers using the top decile function based on the probabilities and decision function values of the classification methods. When α^{3F} and α^{4F} are significant, I can speak of abnormal returns and the higher (more positive) these abnormal returns, the better the market performance of the portfolio constructed. In the Appendix, Table 7 shows more detailed information including the percentage of takeover targets in the portfolio per year, α^{J} and the market risks β. The SVM models use the hyperparameterized values of C and γ as described in Section 4.1. When both indicators are performing well, the portfolio is a proper investment strategy. In Table 3, the total number of takeovers and the average percentage of actual takeover targets per year in the data is shown for both the MD&A and RF sections. In both cases, only 4.3% of all companies are a takeover target the year after a 10-K filing is filed to the SEC.

Based on the first indicator, the best performing model are the ones that use tf-idf document vectorization with either unigrams, bigrams or a combination of the two combined with logistic regression. These 6 models create portfolios with a percentage between 7.1% and 8.1% of takeover targets in the portfolio, which is equal to 16.5% to 18.8% of all takeover targets in the data to be identified in the portfolios. Using logistic regression as classification, the same tf-idf models based on the MD&A section outperform the same tf-idf models based on the Risk Factors section. Next to that, all models combined with logistic regression classification outperform the models with SVM classification based on the percentage of takeover targets in the portfolio. The models based on logistic regression therefore seem most suitable for the identification of takeover targets using large sections of documents like the MD&A and Risk Factors section in 10-Ks. Next to that, the MNB classification method performs better for BoW than tf-idf. This is possibly due to the fact that the document vectors in BoW are actually probabilities of events and tf-idf are weighted by the inverse document frequency (idf), which makes it an alternative to probabilities of events.

The models using one of the two Doc2Vec document vectorization methods do not perform well in terms of the percentage of takeover targets in the portfolio and the percentage of identified takeover targets of all takeover targets in the data. Only the DBOW-HS models based on the MD&A and RF sections and the DM-NS model based on the RF section combined with the logistic classifier have a somewhat high percentage of takeover targets in the portfolio of 6.1%, 6.1% and 6.2% respectively. In contrast to the percentage of takeover targets in the portfolio found in the hyperparameterization process, the Doc2Vec models combined with SVM result in portfolios with low predictive power. 5 of the 8 models even have a lower percentage of takeover targets in the portfolio than the 4.3% takeover targets in the data. These model have a lower predictive power

than portfolios constructed based on random selection. Similar to the percentage of takeover targets in the portfolio lower than 4.3%, when the percentage of identified takeover targets of all takeover targets in the data that are selected in the portfolio is lower than 10%, the models constructed perform worse than random selection. Logically, all models have either a percentage of takeover targets in the portfolio lower than 4.3% and a percentage of identified takeover targets of all takeover targets in the data that are selected in the portfolio lower than 10%, or both higher than these values.

Based on the hyperparameterization results, I expected to achieve better results as, for example, almost all Doc2Vec methods (DM-HS, DM-NS, DBOW-HS and DBOW-NS) based on either the MD&A or RF sections with an SVM classification show more promising results in the hyperparameterization process with more than 7% takeover targets in the portfolios created by the top decile function. Contrary to the expectation, models including tf-idf document vectorization outperform the Doc2Vec. The second indication for a good performing model is the portfolio performance based on the factor models. The market index used for these factor models is the Russell 3000 index. As mentioned, Table 5 shows the results for the annualized returns (α^{3F} and α^{4F}) of the portfolios constructed in this study. The t-value in brackets under the α^{3F} and α^{4F} indicates that the returns of the portfolios, created by choosing the top 10% most likely takeover targets per year for the classification methods, are significantly different from 0. This means that positive significant α^{3F} and $\alpha 4F$ mean the portfolio has positive abnormal returns and outperforms the benchmark index. One can notice that almost all portfolios have positive significant α^{3F} and $\alpha 4F$, which yield positive abnormal returns (alpha's). Regardless of the percentage of takeover targets in the portfolio, the portfolios created outperform the Russell index in most cases. In other words; the positive abnormal returns of the portfolios show that the subset of companies that is identified as a takeover target on annual basis creates a portfolio of significantly better performing stocks than the Russell index without taking into account the percentage of takeover targets in the portfolio. In Table 5, only α^{3F} and α^{4F} are given as the Fama-French Three-Factor and Carhart Four-Factor models both explain more than 90% of the portfolios excess return in contrast to only 70% in Jensen's alpha and CAPM.

The best performing portfolios based on the excess returns solely are the ones with the highest significant α^{3F} and α^{4F}. The best performing portfolio is the one based on the Carhart Four-Factor Model from BoW-2 model based on the MD&A section with SVM classification, which has an annualized abnormal return $\alpha^{4F} = 12.07\%$, which earns more than 12% per year. FF3FM and

Table 5: The percentage of takeover targets in the portfolio (portfolio %) and the percentage of identified takeover targets of all actual takeover targets in the data selected in the portfolio (identified %) created by the top decile function for all models, including estimates of α^{3F} and α^{4F} using monthly time-series data for the returns of the stock in the portfolio. The values for α^{3F} and α^{4F} are annualized and represent the annual excess return. α^{3F} and α^{4F} indicate abnormal returns for the portfolio when $t >= 1.96$ and $t <= -1.96$. No abnormal returns are created by the portfolio when $t < 1.96$ and $t > -1.96$, which is indicated by *. The different classification methods are logistic regression (LR), multinomial Naive Bayes (MNB) and the support vector machine with RBF kernel (SVM).

		MD&A				Risk Factors			
Vectorization	Classification	Portfolio %	Identified %	α^{3F}	α^{4F}	Portfolio %	Identified %	α^{3F}	α^{4F}
BoW-1	LR	5.7	13.3	7.61%	7.62%	5.0	11.7	4.48%	4.78%
				(2.23)	(2.22)			(3.05)	(3.23)
	MNB	5.8	13.5	5.40%	5.87%	5.4	12.6	5.38%	5.73%
				(3.49)	(3.77)			(3.42)	(3.63)
	SVM	4.6	10.7	3.45%	3.75%	3.5	8.2	15.59%*	14.75%*
				(2.20)	(2.37)			(1.32)	(1.25)
BoW-12	LR	5.8	13.5	4.67%	5.03%	5.6	13.1	4.60%	4.82%
				(2.77)	(2.97)			(2.82)	(2.93)
	MNB	5.6	13.0	6.04%	6.62%	5.7	13.3	4.57%	4.98%
				(3.51)	(3.83)			(3.17)	(3.44)
	SVM	5.3	12.4	7.84%	8.24%	5.5	12.9	3.64%	4.12%
				(3.23)	(3.37)			(2.18)	(2.45)
BoW-2	LR	5.6	13.0	7.57%	8.06%	5.8	13.5	17.40%*	16.29%*
				(3.91)	(4.14)			(1.41)	(1.32)
	MNB	6.0	14.0	7.57%	7.79%	6.1	14.3	4.90%	5.04%
				(4.16)	(4.27)			(3.12)	(3.20)
	SVM	5.0	11.6	11.48%	12.07%	3.6	8.4	15.84%*	14.97%*
				(3.58)	(3.74)			(1.34)	(1.27)
tf-idf-1	LR	8.0	18.4	5.14%	5.22%	7.5	17.4	5.44%	5.47%
				(3.42)	(3.46)			(4.34)	(4.34)
	MNB	3.8	8.8	3.01%*	3.19%*	4.4	10.3	4.79%	5.02%
				(1.85)	(1.94)			(3.19)	(3.32)
	SVM	4.6	10.5	6.14%	6.44%	4.8	11.2	9.91%	10.71%
				(3.88)	(4.04)			(1.96)	(2.10)

Continued on next page

Table 5 – continued from previous page

		MD&A				Risk Factors			
Vectorization	Classification	Portfolio %	Identified %	α^{3F}	α^{4F}	Portfolio %	Identified %	α^{3F}	α^{4F}
tf-idf-12	LR	8.1	18.8	6.42% (4.10)	6.63% (4.21)	7.4	17.3	4.79% (3.81)	4.87% (3.85)
tf-idf-12	MNB	3.4	7.8	-0.66%* (-0.44)	-0.09%* (-0.06)	3.1	7.2	4.36%* (1.81)	4.84% (1.99)
	SVM	5.8	13.5	8.41% (3.43)	8.69% (3.52)	5.9	13.9	8.76% (2.76)	9.06% (2.83)
tf-idf-2	LR	7.8	18.1	8.71% (3.51)	9.15% (3.66)	7.1	16.5	6.36% (4.21)	6.63% (4.36)
	MNB	3.3	7.6	2.17%* (1.37)	2.62%* (1.64)	4.1	9.5	4.55% (2.33)	5.17% (2.63)
	SVM	5.2	12.0	10.57% (3.31)	11.34% (3.52)	5.8	13.5	5.92% (3.22)	6.20% (3.35)
DBOW-HS	LR	6.1	14.2	5.69% 3.87	5.84% 3.95	6.1	14.2	3.61% (2.56)	3.79% (2.68)
	SVM	3.2	7.4	11.95%* (0.91)	10.95%* (0.83)	3.2	7.5	14.21%* (1.09)	13.15%* (1.01)
DBOW-NS	LR	5.6	13.0	2.61% (2.11)	2.97% (2.38)	5.3	12.4	5.08% (2.92)	5.55% (3.17)
	SVM	4.9	11.3	4.59%* (1.21)	5.39%* (1.41)	4.9	11.5	4.96% (2.67)	5.33% (2.86)
DM-HS	LR	5.3	12.2	4.36% (2.88)	4.60% (3.02)	5.1	11.9	5.84% (3.74)	6.31% (4.01)
	SVM	3.3	7.7	11.98%* (0.91)	11.12%* (0.85)	3.4	7.9	13.92%* (1.07)	13.07*% (1.01)
DM-NS	LR	4.9	11.9	3.92% (1.99)	4.18% (2.11)	6.2	14.5	10.74% (1.96)	11.12% (2.02)
	SVM	3.2	7.4	11.57%* (0.89)	10.77%* (0.82)	4.8	11.3	3.88% (2.79)	4.19% (2.99)

C4FM generate quite similar results. Therefore, the second highest positive abnormal return is given by the Fama-French Three-Factor Model based on the portfolio created by the same BoW-2 SVM model based on the MD&A section, which yields an annual abnormal return of $\alpha^{4F} = 11.34\%$. Another performing portfolio is the portfolio created by the tf-idf-2 model based on the MD&A section with SVM classification, which yields annual positive abnormal returns of $\alpha^{3F} = 10.57\%$ and $\alpha^{4F} = 11.34\%$ for the FF3FM and C4FM respectively. Furthermore, the portfolio created by the

DM-NS logistic regression model based on the Risk Factors section also yield an annual abnormal return of more than 10% for both the FF3FM and C4FM as $\alpha^{3F} = 10.74\%$ and $\alpha^{4F} = 11.12\%$. Furthermore, portfolios created by the DBOW-HS SVM and DM-HS SVM models based on both the MD&A and RF sections all yield an annual return of more than 11%. However, these excess returns do not yield abnormal returns.

Next to the mentioned portfolios, the best performing portfolio is the one created by the MD&A tf-idf-2 logistic regression model with more than 8% and 9% of annual abnormal return as $\alpha^{3F} = 8.71\%$ and $\alpha^{4F} = 9.15\%$.

Only the MD&A tf-idf-12 MNB model creates a portfolio has negative values as $\alpha^{3F} = -0.66\%$ and $\alpha^{4F} = -0.09\%$. However, these excess returns do not yield abnormal returns.

Looking at the combination of both a higher percentage of takeover targets in the portfolio and the market performance of the portfolio, there is one model that stands out; the tf-idf-2 logistic regression model based on the MD&A section. Two other models also yield interesting results in both areas of interest; the tf-idf-12 logistic regression model based on the MD&A section and the tf-idf-2 logistic regression model based on the Risk Factors section. The tf-idf-2 logistic regression odel based on the MD&A section results in 7.8% of the companies in the portfolio to be a takeover target, which is equal to a total of 230 companies in the portfolio of the 2945 companies which are actually a takeover target. This means that in total 18.1% of all 1271 takeover targets in the MD&A section data are selected in the portfolio. Furthermore, this portfolio yields an annual abnormal return return of over 8% and 9% for the FF3FM and C4FM respectively. The tf-idf-12 logistic regression model based on the MD&A section has a higher percentage of takeover targets in the portfolio at 8.1%, which is equal to a total number of 239 of the 2945 companies in the portfolio that are actually a takeover target. This is equal to a total percentage of 18.8% of the companies that are a takeover target the year after filing a 10-K to the SEC being selected using the top decile function with this model. However, the portfolio yields slightly worse results in terms of outperforming the Russell index in comparison with the MD&A tf-idf-2 logistic regression model with $\alpha^{3F} = 6.42\%$ and $\alpha^{4F} = 6.63\%$. The tf-idf-2 logistic regression model based on the Risk Factors section has a lower percentage of takeover targets in the portfolio with 7.1%, which is equal to a total number of 212 of the 3001 companies being a takeover target. In total, 16.5% of the 1286 companies that actually are a takeover target in the Risk Factors sample are identified in the portfolio.

5 Conclusions

In this thesis, I compare 52 models that classify non-takeover and takeover targets based on textual data in 10-Ks filed by companies listed on either the NASDAQ, NYSE or AMEX. I compare the performance based on the percentage of takeover targets in the portfolio created using the top decile function for every model, and the excess returns of the portfolio with three different factor models. Due to the absence of research that focuses on identifying takeover targets based on textual data, this thesis tries to bridge the gap between predicting takeover targets of public companies and analysing textual data in 10-Ks. A study about changes in 10-Ks provides significant returns of over 22% on annual basis for some models, see Cohen et al. (2016). I study whether takeover target identification using 10-Ks provides similar results.

The first research question is:

> *1. Which combination of document vectorization techniques based on the textual data in 10-Ks and classification methods of these document vectors, if any, is able to best identify takeover targets in the portfolio?*

In this research, I show that the models resulting in the highest percentage of takeover targets in the portfolio are all based on the tf-idf document vectorization method, used with either unigrams, bigrams or the combination of the two, combined with logistic regression classification with a slightly higher percentage of takeover targets in the portfolio for the models using the MD&A section compared to the RF section. BoW and Doc2Vec document vectorization methods in combination with any classification method did not yield results at the same level of the models which used tf-idf with logistic regression. These tf-idf logistic regression models yield a percentage of takeover targets in the portfolio between 7.1% and 8.1%. Next to that, logistic regression performed better than MNB and support vector machines based models, except for models based on BoW document vectorization which yielded a slightly better result using MNB as the classification method. Therefore, I conclude that identifying takeover targets based on textual data in 10-K filings can be classified best using tf-idf document vectorization methods and logistic regression classification.

The second research question is:

> *2. Which portfolio, if any, based on document vectorization techniques and classification methods is best to invest in with regard to outperforming the market and the percentage of takeover targets in the portfolio?*

Almost all of the 52 model combinations create portfolios that are outperforming the benchmark index, the Russell 3000 index, based on the calculated excess return using the Fama French Three-Factor and Carhart Four-Factor Model. This might show that the portfolios created using the top decile function for the probabilities and decision function values of the classification methods are, on average, better performing stocks. However, to answer this question, I need to look at the portfolios that both outperform the Russel index as well as identify a high percentage of takeover targets in the portfolio. Therefore, I look at the tf-idf logistic regression model combinations. From all six models, one portfolio stands out based on both predictive power of identified takeover targets and the portfolio generating positive abnormal returns. This portolio is created using the tf-idf-2 logistic regression model based on the MD&A section. This model creates a portfolio that yields an excess return of more than 8% and 9% on annual basis based on the FF3FM and C4FM respectively. Furthermore, 7.8% of the 2945 companies in this portfolio are takeover targets, which means 230 of the 1271 takeover targets in the data (18.1%) are identified in the portfolio. However, this result does not come close to the annual abnormal returns of 22% described in Cohen et al. (2016).

The expectation that the Doc2Vec document vectorization methods combined with classification methods would provide better results, in terms of both predictive power of identified takeover targets in the portfolio and the portfolio's market performance, than the BoW and tf-idf document vectorization methods is not met. Portfolios created by the MD&A based DM-NS logistic regression model outperform the benchmark index with more than 11%. However, this portfolio does not provide a relatively high percentage of takeover targets in the portfolio created by the top decile function.

Therefore, I conclude that the best investment strategy is based on the tf-idf-12 logistic regression model based on the MD&A section which yields 7.8% of the companies in the portfolio to be an actual takeover target, 18.1% of all actual takeover targets in the data to be identified in the portfolio, and annual returns of more than 8% and 9% based on the Fama-French Three-Factor Model and the Carhart Four-Factor Model.

The last question I want to answer is the following:

> *3. Do classification models based on textual data from 10-Ks perform better than classification models based on numerical characteristics in terms of the percentage of takeover targets in the portfolio and outperforming the market?*

I did find only two papers in the literature that create out-of-sample portfolios with positive abnormal returns and a percentage of takeover targets in the portfolio created of more than 7%. In the first one, based on a sample of Australian public firms, Rodrigues and Stevenson (2013) find portfolios with 15.79%, 22.50% and 33.33% of takeover targets in 2009, 2010 and 2011. However, these portfolios consist of only 19, 40 and 18 companies respectively and the portfolios yield no abnormal returns in 2009, 5.45% annual abnormal return in 2010 and 6.78% annual abnormal return in 2011. Therefore, this paper is not comparable with our results as our portfolio is substantially larger and the abnormal returns are higher at more than 9% annually.

However, Danbolt et al. (2016) creates portfolios based on the top quintile function using logit models with different sets of numerical characteristics and ratios. In contrast with our research, the sample of stocks is based on the London Stock Exchange. The average percentage of companies that are takeover targets in the data is higher at 5.0% compared to the 4.3% of takeover targets in the data in this study. The portfolio constructed by the best performing model has roughly the same size of our portfolios with 3.534 predicted takeover targets and the percentage of takeover targets in the portfolio is 8.52% which is higher than the 7.81% (230 out of 2945) of takeover targets in our best performing portfolio created by the tf-idf-2 logistic regression model based on the MD&A section. However, the portfolio performance of the sample created by Danbolt et al. (2016) only yields abnormal returns when possible bankrupt firms are mitigated. The portfolio yields monthly abnormal returns of 0.9% which is equal to 10.8% annual abnormal returns, where the portfolio created by the tf-idf-2 logistic regression model based on the MD&A section yields an annual abnormal return of 9.15%.

Therefore, I conclude that predicting takeover targets using numerical characteristics of companies probably yield results equal to models that predict takeover targets using textual data based on the percentage of takeover targets in the portfolio and the market performance of the portfolio. Though, predicting takeover targets using numerical characteristics is not studied on American public companies which makes the comparison more difficult. There is a possibility that models using numerical characteristics of American public companies to predict takeover targets perform better or worse than prediction models using textual data of American public companies to predict takeover targets. Further research would be necessary to obtain more solid conclusions.

In this thesis, I use two different sections, the Management's Discussion & Analysis and Risk Factors, from the 10-Ks combined with three different document vectorization methods and three classification methods to identify takeover targets and create portfolios based on the most likely

takeover targets per year from 2010 to 2018. There is one model that creates a portfolio that stands out in both the percentage of takeover targets in the portfolio and outperforming the market. The percentage of takeover targets in the portfolio might indicate that either the MD&A and RF section from the 10-Ks are not that informative about the probability of being a takeover target. However, prediction models of takeover targets based on numerical characteristics of companies do not yield better results than the models used in this study. There might be some methodological choices that restrict the models to generate better results. For example, both BoW and Doc2Vec document vectorization methods are not able to create properly separable document vectors in combination with the classification methods, or the information about the likeliness of being a takeover target is fully incorporated in the price of the stock already. Some pitfalls may be the classification methods being based on the 5 years previous to the year being predicted, the use of words that appear in more than 0.5% of the documents, the number of iterations for the Doc2Vec document classification algorithms to be incorrect for proper classification, or the top decile function generating a too large or too small portfolio of companies to invest in. Further research on identifying takeover targets based on textual data from 10-Ks should focus on tf-idf document vectorization with logistic regression or other classification methods, while creating document vectors with more types of n-grams, different percentages for the number of documents a word to be in and changing parameters not discussed in this research. Furthermore, research that uses classification methods to identify takeover targets based on numerical characteristics of companies could also be revised and updated.

References

Bartov, E. and Konchitchki, Y. (2017). Sec filings, regulatory deadlines, and capital market consequences. *Accounting Horizons*, 31(4):109–131.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Brown, S. V. and Tucker, J. W. (2011). Large-sample evidence on firms' year-over-year MD&A modifications. *Journal of Accounting Research*, 49(2):309–346.

Carhart, M. M. (1997). On persistence in mutual fund performance. *The Journal of finance*, 52(1):57–82.

Cohen, L., Malloy, C., and Nguyen, Q. (2016). Lazy prices. *Harvard Business School, Working Paper*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Danbolt, J., Siganos, A., and Tunyi, A. (2016). Abnormal returns from takeover prediction modelling: challenges and suggested investment strategies. *Journal of Business Finance & Accounting*, 43(1-2):66–97.

Dee, C. C., Hillison, W., and Pacini, C. (2010). No news is bad news: Market reaction to reasons given for late filing of Form 10-K. *Research in Accounting Regulation*, 22(2):121–127.

Dietrich, J. K. and Sorensen, E. (1984). An application of logit analysis to prediction of merger targets. *Journal of Business Research*, 12(3):393–402.

Fama, E. F. and French, K. R. (1992). The cross-section of expected stock returns. *the Journal of Finance*, 47(2):427–465.

Franks, J. R. and Harris, R. S. (1989). Shareholder wealth effects of corporate takeovers: the UK experience 1955–1985. *Journal of financial Economics*, 23(2):225–249.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Jensen, M. C. (1968). The performance of mutual funds in the period 1945-1964. *The Journal of finance*, 23(2):389–416.

Kim, D., Seo, D., Cho, S., and Kang, P. (2019). Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec. *Information Sciences*, 477:15–29.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Li, F. (2008). Annual report readability, current earnings, and earnings persistence. *Journal of Accounting and economics*, 45(2-3):221–247.

Li, F. (2010). The information content of forward-looking statements in corporate filings — A naïve Bayesian machine learning approach. *Journal of Accounting Research*, 48(5):1049–1102.

Lintner, J. (1965). Security prices, risk, and maximal gains from diversification. *The journal of finance*, 20(4):587–615.

Loughran, T. and McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, 66(1):35–65.

Loughran, T. and McDonald, B. (2016). Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Morck, R., Shleifer, A., Vishny, R. W., Horck, R., Shleifer, A., and Vishny, R. W. (1988). Characteristics of Hostile and Friendly Takeovers. In *Corporate Takeovers: Causes and Consequences (University of Chicago Press / NBER.*

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Palepu, K. G. (1986). Predicting takeover targets: A methodological and empirical analysis. *Journal of Accounting and Economics*, 8(1):3–35.

Powell, R. (2004). Takeover Prediction Models and Portfolio Strategies: A Multinomial Approach. *Multinational Finance Journal*, 8.

Rodrigues, B. D. and Stevenson, M. J. (2013). Takeover prediction using forecast combinations. *International Journal of Forecasting*, 29(4):628–641.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442.

Stevens, D. L. (1973). Financial characteristics of merged firms: A multivariate analysis. *Journal of Financial and Quantitative Analysis*, 8(2):149–158.

Ur-Rahman, N. and Harding, J. A. (2012). Textual data mining for industrial knowledge management and text classification: A business oriented approach. *Expert Systems with Applications*, 39(5):4729–4739.

Yazdani, S. F., Murad, M. A. A., Sharef, N. M., Singh, Y. P., and Latiff, A. R. A. (2017). Sentiment classification of financial news using statistical features. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(03):1750006.

Appendix

Historical background

Regulation and SEC history

Historically, only the rich were able to compete on the stock market due to the high risks involved in investing. Due to the level of fraud in the stock market, the first steps to regulation in the United States were the Blue Sky Laws first enacted in 1911. These state regulations were established to protect investors for worthless and fraudulent securities. Blue Sky Laws require sellers to register all stock offerings, sales and trades. Though, the Blue Sky Laws were not enforced properly in their state as companies could still sell stocks at unfair terms if they informed investors about it. Furthermore, state regulations contributed to selling to investors in other states as there were no requirements of full disclosure to these out-of-state investors.

On October 29th 1929, Black Monday was one of the first events leading to the Great Depression. In the years before this crash, sellers, owners, and bankers drove up the prices of securities to get more return by trading the stocks between each other before making the stocks available for the public. As prices were exceptionally high and many customers did not know the actual worth of their assets, the economy was hit hard and the government had to help to solve the problems created by the stock market. A major reason for the government to intervene was the fact that the Federal Reserve (Fed) did not lower their interest rates which resulted in the bankruptcy of numerous traders. Furthermore, the increased money supply before the Great Depression by the Fed was one of the reasons the stock market collapsed. The government decided to help to solve the stock market problems in exchange for government regulations and enforcement. The response from President Roosevelt and his government resulted in Congress passing two major acts, the Securities Act in 1933 and the Securities Exchange Act of 1934, and the foundation of the U.S. Securities and Exchange Commission (SEC) based on Section 4 of the Securities Exchange Act of 1934. The Securities Act, also called the Truth in Securities law, aims on regulating sales of securities between states at the federal level and requires sellers to provide disclosure and information for securities being offered publicly to investors. The Securities Exchange Act governs the secondary trading of securities. The primary market deals with the sale of securities of companies to investors directly, and the secondary market refers to a financial market where stocks, bonds, options and futures can be traded, e.g. the NYSE, AMEX and Nasdaq.

The SEC regulates the Securities Exchange Act, and many more acts that were passed in the years following the pass of the Act of 1934. The SEC therefore received a considerable amount of power in the stock market which resulted in companies who are trading their securities on secondary markets filing file certain documents to the SEC in accordance with strict reporting schedules. The Securities Exchange Act refers to the Form 10-K as the annual report pursuant to sections 13 or 15(d) 1934[9] and these are filed since the start of the SEC by publicly traded companies. Next to the strict reporting schemes, the SEC was allowed to file charges against companies and individuals who violated the rules and regulations defined by the SEC. These changes paved the way for reluctant investors to go back to investing in the stock market and boosting the economy after the Second World War. Furthermore, investors could request information before investing in securities traded.

EDGAR, Form 10-K and MD&A

Since 1994, most filings are filed using the online Electronic Data Gathering, Analysis, and Retrieval system (EDGAR) of the SEC. EDGAR is available to the public and contains data from 1993 until today. Nowadays, around 3,000 filings are processed by the system daily and the system makes more than 3,000 terabytes of data available to the public on annual basis.

In the EDGAR database, the Form 10-K is one of the most informative filings to get a better grasp of a company. A 10-K contains useful information for investors and consists of more information and deeper analysis than annual reports. Therefore, 10-Ks are generally very extensive and long, and reading them takes a vast amount of time. Investors usually only read the most important parts to better understand the risks, opportunities and business operations.

Furthermore, it is important for companies and investors to file their 10-Ks on time. When companies cannot file within the deadline, the company must file a non-timely, NT, filing. Dee et al. (2010) showed companies that announce or file NT filings cause significant negative abnormal returns on stocks. Announcements of late filings are considered to signal bad performance of the company, where the bad performance is not totally reflected in the stock prices at the time the announcements of late filings are made Bartov and Konchitchki (2017). In 2004, the SEC changed the policy regarding the deadlines of the submission of the 10-K in a Final Rule. The 90 day deadline after the fiscal year's end was changed to 60 days for large accelerated filers (companies with a public float (shares held by public investors) value of more than $ 700 million), 75 days for accelerated filers (companies with a public float value between $ 75 and $700 million) and 90 days

[9] `https://www.sec.gov/files/form10-k.pdf`

for non-accelerated filers (companies with a public float value of less than $ 75 million)[10].

Every 10-K consists of four parts containing 15 items, see Table 6. There are no general names for the four parts, but, in general, all forms use the same names for the items. Item 7: Management's Discussion and Analysis of the Financial Condition (MD&A) found in Part II, is arguably the most important part of the 10-Ks as management discusses the companies performance. The

[10] `https://www.sec.gov/fast-answers/answers-form10khtm.html`

Table 6: The structure of a 10-K (as required by the SEC) with Item 1A: Risk Factors and Item 7: Management's Discussion and Analysis highlighted.

	Part I
Item 1	Business
Item 1A	**Risk Factors**
Item 1B	Unresolved Staff Comments
Item 2	Properties
Item 3	Legal Proceedings
Item 4	Mine Safety Disclosures
	Part II
Item 5	Market for Registrant's Common Equity, Related Stockholder Matters and Issuer Purchases of Equity Securities
Item 6	Consolidated Financial Data
Item 7	**Management's Discussion and Analysis (of Financial Condition and Results of Operations)**
Item 7A	Quantitative and Qualitative Disclosures about Market Risks
Item 8	Financial Statements
Item 9	Changes in and Disagreements with Accountants on Accounting and Financial Disclosure
Item 9A	Controls and Procedures
Item 9B	Other Information
	Part III
Item 10	Directors, Executive Officers and Corporate Governance
Item 11	Executive Compensation
Item 12	Security Ownership of Certain Beneficial Owners and Management and Related Stockholder Matters
Item 13	Certain Relationships and Related Transactions, and Director Independence
Item 14	Principal Accountant Fees and Services
	Part IV
Item 15	Exhibits, Financial Statement Schedules

performance of the company is analyzed using both qualitative and quantitative measures, can discuss compliance, risks, future goals and plans, and reflects the opinion and thoughts of the management based on its view on the company. Though, the MD&A needs to meet certain standards and should discuss both positive and negative points in the business, according to the Financial Accounting Standards Board (FASB), this section of the 10-Ks might reflect the possibilities of the company at best and is used for tone analyses in the literature, see e.g. Loughran and McDonald (2011).

Word2Vec derivations

Continuous Bag of Words Model Updates

Let the vocabulary size consist of V words and let the dimension of every word vector be N. Then for every word in the text there are C words in the context, (at most) $\frac{1}{2}C$ words before and (at most) $\frac{1}{2}C$ words after. The C words are represented by $\{w_{c,1}, \ldots, w_{c,C}\}$. Only for the first and the last few words in the text the number of words C in the context is smaller. The input vectors for word w are equal to the one-hot encoded vectors of the C words in the context, which means

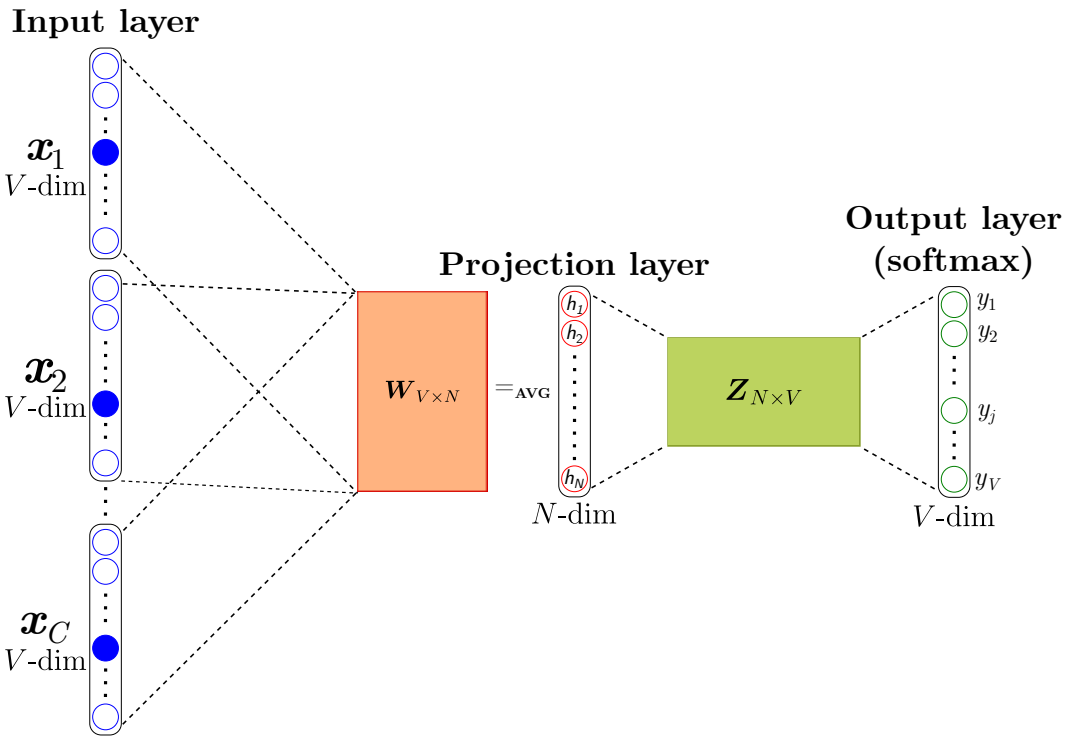

Figure 8: Architecture of the Continuous Bag of Words Model.

that every word in the context is equal to zero vectors with size V equal to the vocabulary size and a 1 at the position of that word in the vocabulary V, also called one-hot encoded vectors. The one-hot encoded vector of word $w_{c,l}$ for $l = 1, \ldots, C$ is represented by $\boldsymbol{x}_l$. Then, $V \times N$ matrix $\boldsymbol{W}$ represents the weights of the single projection layer of the neural network. The j-th row of matrix $\boldsymbol{W}$ contains the N-dimensional word vector $\boldsymbol{v}_j$ of the j-th word of the vocabulary. That means that if word $w_{c,l}$ is at the k-th position in the vocabulary, and $w_{c,l,k} = 1$ and $w_{c,l,k'} = 0$ for $k' \neq k$, word $w_{c,l}$ is represented by $\boldsymbol{v}_{w_{c,l}} = \boldsymbol{v}_k$. Then, the projection layer in the neural network can be updated by

$$\boldsymbol{h} = \frac{1}{C}\boldsymbol{W}^T(\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_C) \tag{26}$$

$$= \frac{1}{C}(\boldsymbol{W}^T\boldsymbol{x}_1 + \cdots + \boldsymbol{W}^T\boldsymbol{x}_C) \tag{27}$$

$$= \frac{1}{C}(\boldsymbol{v}_{w_{c,1}} + \cdots + \boldsymbol{v}_{w_{c,C}})^T, \tag{28}$$

where $\boldsymbol{h}$ is a vector of lenght N (Rong, 2014). This layer is called the projection layer as the activation of the layer is linear, only neural networks with non-linear activation functions have so-called hidden layers. In this case, the activation is linear as matrix $\boldsymbol{W}$ is the activation function and linear. $N \times V$ matrix $\boldsymbol{Z}$, in the literature also referenced to as $\boldsymbol{W}'$ which is not the transpose of $\boldsymbol{W}$ and therefore somewhat confusing, is another weight matrix used for the projection to the output layer where column j of matrix $\boldsymbol{Z}$ contains vector $\boldsymbol{u}_{w_j}$. For every word in the vocabulary a score can be computed by

$$s_j = \boldsymbol{u}_{w_j}^T\boldsymbol{h} \tag{29}$$

$$= \frac{1}{C}\boldsymbol{u}_{w_j}^T(\boldsymbol{v}_{w_{c,1}} + \cdots + \boldsymbol{v}_{w_{c,C}})^T \tag{30}$$

$$= \frac{1}{C}(\boldsymbol{u}_{w_j}^T\boldsymbol{v}_{w_{c,1}}^T + \cdots + \boldsymbol{u}_{w_j}^T\boldsymbol{v}_{w_{c,C}}^T), \tag{31}$$

and then softmax is used to approximate the posterior distribution for every word in the vocabulary with

$$p(w_j|w_{c,1}, \ldots, w_{c.C}) = y_j = \frac{\exp(s_j)}{\sum_{i=1}^{V}\exp(s_i)}, \tag{32}$$

where the j-th unit in the output layer is y_j. The softmax function in (32) predicts exactly one output layer to be activated. In Figure 8, a visualization of the architecture of CBOW is presented. One can see that the input layer consists of C V-dimensional vectors representing the context input words, which are all multiplied with matrix $\boldsymbol{W}$ and averaged to produce the projection layer update

vector $\boldsymbol{h}$. Then this vector is multiplied with matrix $\boldsymbol{Z}$ to produce scores which are then converted to the softmax output layer $\boldsymbol{y}$.

Both weighing matrix $\boldsymbol{W}$ and $\boldsymbol{Z}$ are updated and tuned using stochastic gradient descent (SGD) and backpropagation (Rumelhart et al., 1986). The loss function is derived with regard to the actual word w_0 that should be generated as output. The word w_0 is surrounded by C context words in the actual text. Given C context words, the actual word w_0 is the best prediction for the neural network and can be found at index $j*$ in $\boldsymbol{y}$. $\boldsymbol{y}$ is the vector of the probability of every word being the word that is surrounded by the context words after softmax. This results in maximizing the probability that w_O is the predicted word given C context words. Then, the loss function is minimized by

$$L = -\log p(w_O|w_{c,1}, \dots w_{c,C}) \tag{33}$$

$$= -\log y_{j*} \tag{34}$$

$$= -\log \Big(\frac{\exp(s_j*)}{\sum_{i=1}^{V} \exp(s_i)}\Big) \tag{35}$$

$$= -s_{j*} + \log \sum_{i=1}^{V} \exp(s_i), \tag{36}$$

which is equivalent to maximizing $p(w_0|w_{c,1}, \dots, w_{c,C})$.

As the weight matrices $\boldsymbol{W}$ and $\boldsymbol{Z}$ are updated using stochastic gradient descent (SGD) which requires the derivatives of the loss function $L$with respect to $\boldsymbol{W}$ and $\boldsymbol{Z}$. The loss function L can be written in terms of $\boldsymbol{W}$ and $\boldsymbol{Z}$ which depend on $\boldsymbol{s}$ by

$$L = L(s_1(\boldsymbol{W}, \boldsymbol{Z}), \dots, s_V(\boldsymbol{W}, \boldsymbol{Z})), \tag{37}$$

which results in the following derivative with respect to Z_{ij}

$$\frac{\partial L}{\partial Z_{ij}} = \sum_{k=1}^{V} \frac{\partial L}{\partial s_k} \frac{\partial s_k}{\partial Z_{ij}}, \tag{38}$$

where Z_{ij}, the element in column i and row j of matrix $\boldsymbol{Z}$, connects h_i of the projection layer to y_j of the output layer. Therefore, all derivatives are zero apart from when $k = j$ in (38). This results in

$$\frac{\partial L}{\partial Z_{ij}} = \frac{\partial L}{\partial s_j} \frac{\partial s_j}{\partial Z_{ij}}. \tag{39}$$

Taking the derivative with respect to s_j from L, I obtain

$$\frac{\partial L}{\partial s_j} = \frac{\partial(-s_{j*} + \log \sum_{i=1}^{V} \exp(s_i))}{\partial s_j} \tag{40}$$

$$= -\mathbb{1}(j = j*) + y_j = \varepsilon_j, \tag{41}$$

where $\mathbb{1}(j = j*)$ is equal to 1 if $j = j*$, and zero otherwise, and the j-th element of the output layer has prediction error ε_j. The derivative with regard to Z_{ij} is derived by

$$\frac{\partial s_j}{\partial Z_{ij}} = \frac{\partial \boldsymbol{u}_{w_j}^T \boldsymbol{h}}{\partial Z_{ij}} = h_i, \tag{42}$$

where h_i equals the i-th node of the projection layer. Therefore, (39) results in

$$\frac{\partial L}{\partial Z_{ij}} = \frac{\partial L}{\partial s_j} \frac{\partial s_j}{\partial Z_{ij}} = \varepsilon_j h_i. \tag{43}$$

Then, SGD results in the following update for every column i and row j in matrix $\boldsymbol{Z}$:

$$Z_{ij}^{(new)} = Z_{ij}^{(old)} - \eta \frac{\partial L}{\partial Z_{ij}} = Z_{ij}^{(old)} - \eta \varepsilon_j h_i, \tag{44}$$

where η is the step size, also called learning rate in the context of machine learning. At last, the update for every column of matrix $\boldsymbol{Z}$, where column j of matrix $\boldsymbol{Z}$ is defined as $\boldsymbol{u}_{w_j}$, is given by

$$\boldsymbol{u}_{w_j}^{(new)} = \boldsymbol{u}_{w_j}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{u}_{w_j}} = \boldsymbol{u}_{w_j}^{(old)} - \eta \varepsilon_j \boldsymbol{h}. \tag{45}$$

Now, the SGD update for matrix $\boldsymbol{W}$ is needed. The derivative of L with respect to W_{ki} can be rewritten by

$$\frac{\partial L}{\partial W_{ki}} = \sum_{j=1}^{N} \frac{\partial L}{\partial h_j} \frac{\partial h_j}{\partial W_{ki}} = \frac{\partial L}{\partial h_i} \frac{\partial h_i}{\partial W_{ki}}, \tag{46}$$

where W_{ki} is the element of column k and row i in $V \times N$ matrix $\boldsymbol{W}$ and h_i again the i-th node in the projection layer. Remember than W_{ki} connects node k of the input to node i of the projection layer. Then,

$$\frac{\partial L}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial L}{\partial s_j} \frac{\partial s_j}{\partial h_i} = \sum_{j=1}^{V} \varepsilon_j Z_{ij}, \tag{47}$$

as

$$\frac{\partial s_j}{\partial h_i} = \frac{\partial \boldsymbol{u}_{w_j}^T \boldsymbol{h}}{\partial h_i} = Z_{ij}, \tag{48}$$

and

$$\frac{\partial h_i}{\partial W_{ki}} = \frac{1}{C} x_{i,w_{c,l}} \quad \text{for } l = 1, \ldots, C, \tag{49}$$

which results in

$$\frac{\partial L}{\partial W_{ki}} = \frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij} x_{i,w_{c,l}} \quad \text{for } l = 1,\ldots,C, \tag{50}$$

where x_{i,w_c} is node i of input word w_c of the C context words. For all C words this results in the following update for every element in matrix $\boldsymbol{W}$:

$$W_{ki}^{(new)} = W_{ki}^{(old)} - \eta\frac{\partial L}{\partial W_{ki}} = W_{ki}^{(old)} - \eta\frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij} x_{i,w_{c,l}} \quad \text{for } l = 1,\ldots,C, \tag{51}$$

where η is again the step size or learning rate. Then, every row of matrix $\boldsymbol{W}$ is updated for all C words in the context by

$$\boldsymbol{v}_{w_{c,l}}^{(new)} = \boldsymbol{v}_{w_{c,l}}^{(old)} - \eta\frac{\partial L}{\partial \boldsymbol{v}_{w_{c,l}}} = \boldsymbol{v}_{w_{c,l}}^{(old)} - \eta\frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij}\boldsymbol{x}_{w_{c,l}} \quad \text{for } l = 1,\ldots,C. \tag{52}$$

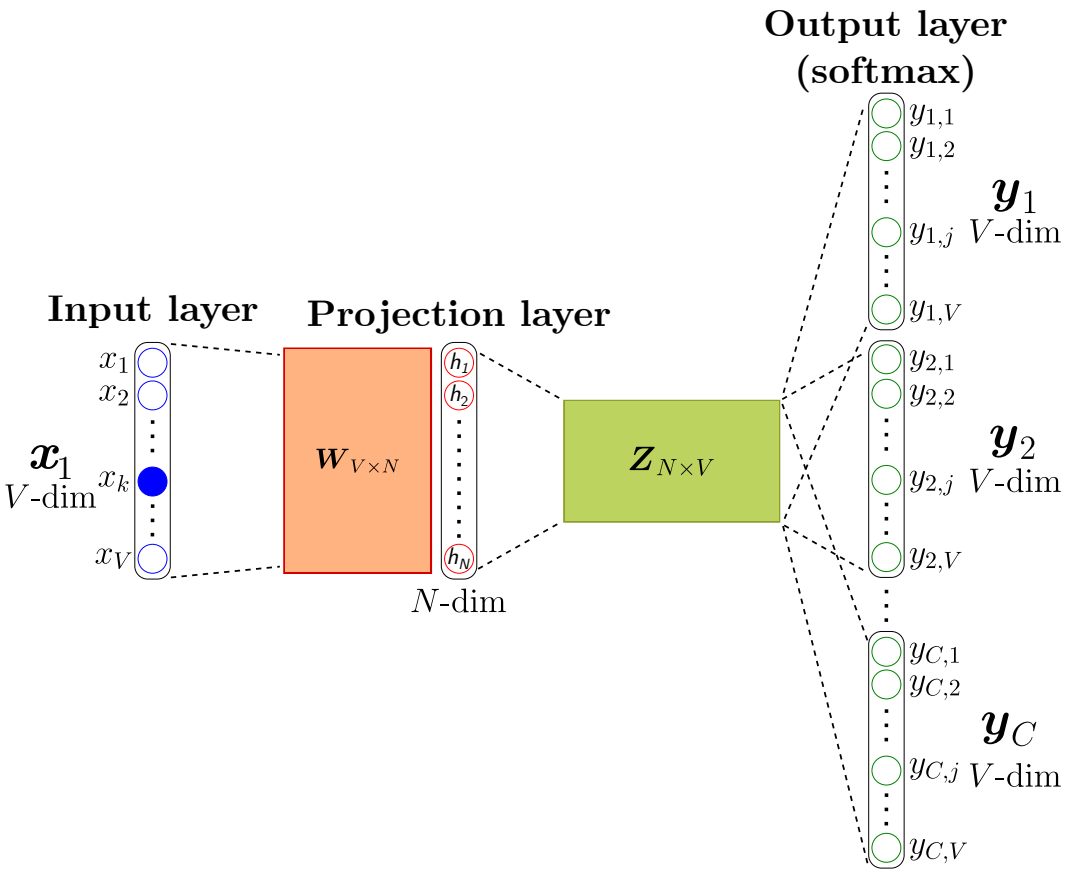

Figure 9: Architecture of the Continuous Skip-gram Model.

Continuous Skip-gram Model Updates

The Continuous Skip-gram Model (SG) uses the input layer for the center word w_I which predicts C objective context words defined as $\{w_{c,1}, \ldots, w_{c,C}\}$ in the output layer. Figure 9 shows the architecture of SG.

Similar to Continuous Bag of Words, the projection layer in the neural network can be updated by the linear activation matrix $\boldsymbol{W}$. With SG, only one word w_I is in the input layer. Therefore, the projection layer can be updated by

$$\boldsymbol{h} = \boldsymbol{W}^T \boldsymbol{x}_1 \tag{53}$$

$$= \boldsymbol{v}_{w_I}^T, \tag{54}$$

where $\boldsymbol{x}_1$ is the one-hot encoded vector belonging to input center word w_I and $\boldsymbol{v}_{w_I}$ is the N-dimensional word vector from weight matrix $\boldsymbol{W}$ of the word w_I. Again, column j of the weight matrix $\boldsymbol{Z}$ that connects the projection layer with the output layer is equal to vector u_{w_j}. Therefore, for all C output context words the score of every word j in the vocabulary can be defined by

$$s_{j,l} = \boldsymbol{u}_{w_{c,l}}^T \boldsymbol{h} \tag{55}$$

$$= \boldsymbol{u}_{w_{c,l}}^T \boldsymbol{v}_{w_I}^T \quad \text{for } l = 1, \ldots, C. \tag{56}$$

As the weight matrix Z is the same for all output words, $s_{j,l}$ is equal for all C output words.

Then for all C context words the V-dimensional $\boldsymbol{y}_l$ vectors can be derived from

$$p(w_{c,l}|w_I) = y_{j,l} = \frac{\exp(s_{j,l})}{\sum_{i=1}^{V} \exp(s_{i,l})} \quad \text{for } l = 1, \ldots, C, \tag{57}$$

where $s_{j,1} = \cdots = s_{j,C}$ which results in $y_{j,1} = \cdots = y_{j,C}$ and thus $\boldsymbol{y}_1 = \cdots = \boldsymbol{y}_C$.

Maximizing the probability that the context output words given the input word w_I, $p(w_{c,1}, \ldots, w_{c,C}|w_I)$, results in the loss function to be minimized

$$L = -\log p(w_{c,1}, \ldots, w_{c,C}|w_I) \tag{58}$$

$$= -\log \prod_{l=1}^{C} p(w_{c,l}|w_I) \tag{59}$$

$$= -\log \prod_{l=1}^{C} \Big(\frac{\exp(s_{j^*,l})}{\sum_{i=1}^{V} \exp(s_{i,l})} \Big) \tag{60}$$

$$= -\sum_{l=1}^{C} s_{j*,l} + \sum_{l=1}^{C} \log \sum_{i=1}^{V} \exp(s_{i,l}), \tag{61}$$

where $s_{j^*,l}$ is the score of the actual context output word w_l at index j^* in $\boldsymbol{y}_l$ which is not equal for the C output context words. Similar to CBOW, the loss function is written in terms of the weight matrices $\boldsymbol{W}$ and $\boldsymbol{Z}$ depending on $\boldsymbol{u}_l$ for $l = 1, \ldots, C$ by

$$L = L(s_{1,1}(\boldsymbol{W}, \boldsymbol{Z}), s_{2,1}(\boldsymbol{W}, \boldsymbol{Z}), \ldots, s_{V-1,C}(\boldsymbol{W}, \boldsymbol{Z}), s_{V,C}(\boldsymbol{W}, \boldsymbol{Z})). \tag{62}$$

Stochastic gradient descent (SGP) is used to update $\boldsymbol{W}$ and $\boldsymbol{Z}$. This results in:

$$\frac{\partial L}{\partial Z_{ij}} = \sum_{k=1}^{V} \sum_{l=1}^{C} \frac{\partial L}{\partial s_{k,l}} \frac{\partial s_{k,l}}{\partial Z_{ij}}, \tag{63}$$

where Z_{ij} connects node i of the projection layer with every index j of the output layer of the C output context words. From (38), (39), (41) and (42), it follows that

$$\begin{aligned}
\frac{\partial L}{\partial Z_{ij}} &= \sum_{l=1}^{C} \frac{\partial L}{\partial s_{j,l}} \frac{\partial s_{j,l}}{\partial Z_{ij}} && (64)\\
&= \sum_{l=1}^{C} (-\mathbb{1}(j = j^*)_l + y_{j,l}) \frac{\partial s_{j,l}}{\partial Z_{ij}} && (65)\\
&= \sum_{l=1}^{C} (-\mathbb{1}(j = j^*)_l + y_{j,l}) h_i && (66)\\
&= \sum_{l=1}^{C} \varepsilon_{j,l} h_i, && (67)
\end{aligned}$$

where $\mathbb{1}(j = j^*)_l$ is equal to 1 for context word $w_{c,l}$ if the actual context word is at index j^* of $\boldsymbol{y}_l$, and zero otherwise. The SGD update then results in:

$$Z_{ij}^{(new)} = Z_{ij}^{(old)} - \eta \frac{\partial L}{\partial Z_{ij}} = Z_{ij}^{(old)} - \eta \sum_{l=1}^{C} \varepsilon_{j,l} h_i, \tag{68}$$

which can be written as

$$\boldsymbol{u}_{w_j}^{(new)} = \boldsymbol{u}_{w_j}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{u}_{w_j}} = \boldsymbol{u}_{w_j}^{(old)} - \eta \sum_{l=j}^{C} \varepsilon_{j,l} \boldsymbol{h} \tag{69}$$

for every word w_j in the vocabulary that is connected to column $\boldsymbol{u}_{w_j}$ in matrix $\boldsymbol{Z}$, where η is the learning rate.

The SGD update of every element of matrix $\boldsymbol{W}$ can be derived by taking the derivative of the loss function with respect to W_{ki} which has the same formula as (46). The derivative of the loss function with respect to h_i can be derived by

$$\frac{\partial L}{\partial h_i} = \sum_{j=1}^{V} \sum_{l=1}^{C} \frac{\partial L}{\partial s_{j,l}} \frac{\partial s_{j,l}}{\partial h_i} = \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij}, \tag{70}$$

as

$$\frac{\partial s_{j,l}}{\partial h_i} = \frac{\partial \boldsymbol{u}_{w_j}^T \boldsymbol{h}}{\partial h_i} = Z_{ij}, \tag{71}$$

and

$$\frac{\partial h_i}{\partial W_{ki}} = x_i \tag{72}$$

which results in

$$\frac{\partial L}{\partial W_{ki}} = \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij} x_i. \tag{73}$$

Then, the updated value retrieved by backpropagation and SDG for every element of matrix $\boldsymbol{W}$ is obtained by

$$W_{ki}^{(new)} = W_{ki}^{(old)} - \eta \frac{\partial L}{\partial W_{ki}} = W_{ki}^{(old)} - \eta \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij} x_i, \tag{74}$$

where η is the step size or learning rate. Then, every row of matrix $\boldsymbol{W}$ is updated by

$$\boldsymbol{v}_{w_I}^{(new)} = \boldsymbol{v}_{w_I}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{v}_{w_I}} = \boldsymbol{v}_{w_I}^{(old)} - \eta \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij} \boldsymbol{x}. \tag{75}$$

Doc2Vec derivations

Distributed Memory Model Updates

Let the vocabulary size consist of V words and P documents and let N be the dimension of every word and document vector after Doc2Vec. For every word in the text, there are C words and document d_I in the context, (at most) $\frac{1}{2}C$ words before and (at most) $\frac{1}{2}C$ words after. The C words are represented by $\{w_{c,1}, \ldots, w_{c,C}\}$. Only for the first and the last few words in the text the number of words C in the context is smaller. The input vectors for word w are equal to the one-hot encoded vectors of the C words in the context, which means that every word in the context is equal to zero vectors with size V equal to the vocabulary size and a 1 at the position of that word in the vocabulary V, also called one-hot encoded vectors. The same holds for the input vector of document d_I which is equal to a one-hot encoded vector $\boldsymbol{b}_1$ The one-hot encoded vector of word $w_{c,l}$ for $l = 1, \ldots, C$ is represented by $\boldsymbol{x}_l$. Then, $V \times N$ matrix $\boldsymbol{W}$ represents the weight matrix of the input words and $P \times N$ matrix $\boldsymbol{D}$ the weight matrix of the document. The combination of these weights represent the weighs of the single projection layer of the neural network. The j-th row of matrix $\boldsymbol{W}$ contains the N-dimensional word vector $\boldsymbol{v}_j$ of the j-th word of the vocabulary. That means that if word $w_{c,l}$ is at the k-th position in the vocabulary, and $w_{c,l,k} = 1$ and $w_{c,l,k'} = 0$

for $k' \neq k$, word $w_{c,l}$ is represented by $\boldsymbol{v}_{w_{c,l}} = \boldsymbol{v}_k$. The p-th row of matrix $\boldsymbol{D}$ contains the N-dimensional document vector $\boldsymbol{t}_p$ of the p-th word of the vocabulary, where $\boldsymbol{t}_{d_I} = \boldsymbol{t}_p$ if document d_I is at the p-th position in the input vector $\boldsymbol{b}_1$ The N-dimensional projection layer vector $\boldsymbol{h}$ is updated by summing or averaging the average update of the C context words one-hot encoded vectors and the one-hot encoded document vector $\boldsymbol{b}_1$. Then, the projection layer in the neural network is updated by

$$\boldsymbol{h} = \frac{1}{C}\boldsymbol{W}^T(\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_C) + \boldsymbol{D}^T\boldsymbol{b}_1 \tag{76}$$

$$= \frac{1}{C}(\boldsymbol{W}^T\boldsymbol{x}_1 + \cdots + \boldsymbol{W}^T\boldsymbol{x}_C) + \boldsymbol{D}^T\boldsymbol{b}_1 \tag{77}$$

$$= \frac{1}{C}(\boldsymbol{v}_{w_{c,1}} + \cdots + \boldsymbol{v}_{w_{c,C}})^T + \boldsymbol{t}_{d_I}, \tag{78}$$

This layer is called the projection layer as the activation of the layer is linear, only neural networks with non-linear activation functions have so-called hidden layers. In this case, the activation is linear as matrix $\boldsymbol{W}$ and $\boldsymbol{D}$ are part of the activation function and linear. $N \times V$ matrix $\boldsymbol{Z}$, in the literature also referenced to as $\boldsymbol{W}'$ which is not the transpose of $\boldsymbol{W}$ and therefore somewhat confusing, is another weight matrix used for the projection to the output layer where column j of matrix $\boldsymbol{Z}$ contains vector $\boldsymbol{u}_{w_j}$. For every word in the vocabulary a score is computed by

$$s_j = \boldsymbol{u}_{w_j}^T\boldsymbol{h} \tag{79}$$

$$= \frac{1}{C}\boldsymbol{u}_{w_j}^T(\boldsymbol{v}_{w_{c,1}} + \cdots + \boldsymbol{v}_{w_{c,C}})^T + \boldsymbol{u}_{w_j}^T\boldsymbol{t}_{d_I} \tag{80}$$

$$= \frac{1}{C}(\boldsymbol{u}_{w_j}^T\boldsymbol{v}_{w_{c,1}}^T + \cdots + \boldsymbol{u}_{w_j}^T\boldsymbol{v}_{w_{c,C}}^T) + \boldsymbol{u}_{w_j}^T\boldsymbol{t}_{d_I}, \tag{81}$$

and then softmax is used to approximate the posterior distribution for every word in the vocabularywith

$$p(w_j|w_{c,1}, \ldots, w_{c.C}, d_I) = y_j = \frac{\exp(s_j)}{\sum_{i=1}^{V}\exp(s_i)}, \tag{82}$$

where the j-th unit in the output layer is y_j. The softmax function in (82) predicts exactly one output layer to be activated. In Figure 4, a visualization of the architecture of DM is presented. One can see that the input layer consists of C V-dimensional vectors representing the context input words, which are all multiplied with matrix $\boldsymbol{W}$ and averaged, and concatenated with matrix $\boldsymbol{D}$ to get the projection layer update vector $\boldsymbol{h}$. Then $\boldsymbol{h}$ is multiplied with matrix $\boldsymbol{Z}$ to produce scores which are then converted to the softmax output layer $\boldsymbol{y}$.

Both weighing matrices $\boldsymbol{D}$, $\boldsymbol{W}$ and $\boldsymbol{Z}$ need to be updated and tuned using stochastic gradient descent (SGD) and backpropagation (Rumelhart et al., 1986). The loss function is derived with

regard to the actual word w_0 that should be generated as output. The word w_0 is surrounded by the C context words and document d_I in the actual text. Given the C context words and document d_I, the actual word w_0 is the best prediction for the neural network and can be found at index $j*$ in $\boldsymbol{y}$. $\boldsymbol{y}$ is the vector of the probability of every word being the word that is surrounded by the context words and the document after softmax. This results in maximizing the probability that w_O is the predicted word given C context words. Then, the loss function is minimized by

$$L = -\log p(w_O|w_{c,1}, \dots w_{c,C}, d_I) \tag{83}$$

$$= -\log y_{j*} \tag{84}$$

$$= -\log \Big(\frac{\exp(s_j*)}{\sum_{i=1}^{V} \exp(s_i)} \Big) \tag{85}$$

$$= -s_{j*} + \log \sum_{i=1}^{V} \exp(s_i), \tag{86}$$

which is equivalent to maximizing $p(w_0|w_{c,1}, \dots, w_{c,C}, d_I)$.

As the weight matrices $\boldsymbol{D}$, $\boldsymbol{W}$ and $\boldsymbol{Z}$ are updated using stochastic gradient descent (SGD) which requires $\partial L/\partial \boldsymbol{D}$, $\partial L/\partial \boldsymbol{W}$ and $\partial L/\partial \boldsymbol{Z}$. The loss function L can be written in terms of $\boldsymbol{D}$, $\boldsymbol{W}$ and $\boldsymbol{Z}$ which depend on $\boldsymbol{s}$ by

$$L = L(s_1(\boldsymbol{D}, \boldsymbol{W}, \boldsymbol{Z}), \dots, s_V(\boldsymbol{D}, \boldsymbol{W}, \boldsymbol{Z})), \tag{87}$$

which results in the following derivative with respect to Z_{ij}

$$\frac{\partial L}{\partial Z_{ij}} = \sum_{k=1}^{V} \frac{\partial L}{\partial s_k} \frac{\partial s_k}{\partial Z_{ij}}, \tag{88}$$

where Z_{ij}, the element in column i and row j of matrix $\boldsymbol{Z}$, connects h_i of the projection layer to y_j of the output layer. Therefore, all derivatives are zero apart from when $k = j$ in (88). This results in

$$\frac{\partial L}{\partial Z_{ij}} = \frac{\partial L}{\partial s_j} \frac{\partial s_j}{\partial Z_{ij}}. \tag{89}$$

Taking the derivative with respect to s_j from L, I obtain

$$\frac{\partial L}{\partial s_j} = \frac{\partial(-s_{j*} + \log \sum_{i=1}^{V} \exp(s_i))}{\partial s_j} \tag{90}$$

$$= -\mathbb{1}(j = j*) + y_j := \varepsilon_j, \tag{91}$$

where $\mathbb{1}(j = j*)$ is equal to 1 if $j = j*$, and zero otherwise, and the j-th element of the output layer has prediction error ε_j. The derivative of with regard to Z_{ij} is derived by

$$\frac{\partial s_j}{\partial Z_{ij}} = \frac{\partial \boldsymbol{u}_{w_j}^T \boldsymbol{h}}{\partial Z_{ij}} = h_i, \tag{92}$$

where h_i equals the i-th node of the projection layer. Therefore, (89) results in

$$\frac{\partial L}{\partial Z_{ij}} = \frac{\partial L}{\partial s_j}\frac{\partial s_j}{\partial Z_{ij}} = \varepsilon_j h_i. \tag{93}$$

This results in the following update for every column i and row j in matrix $\boldsymbol{Z}$:

$$Z_{ij}^{(new)} = Z_{ij}^{(old)} - \eta\frac{\partial L}{\partial Z_{ij}} = Z_{ij}^{(old)} - \eta\varepsilon_j h_i, \tag{94}$$

where η is the step size, also called learning rate in the context of machine learning. At last, the update for every column of matrix $\boldsymbol{Z}$, where column j of matrix $\boldsymbol{Z}$ is defined as $\boldsymbol{u}_{w_j}$, is given by

$$\boldsymbol{u}_{w_j}^{(new)} = \boldsymbol{u}_{w_j}^{(old)} - \eta\frac{\partial L}{\partial \boldsymbol{u}_{w_j}} = \boldsymbol{u}_{w_j}^{(old)} - \eta\varepsilon_j \boldsymbol{h}. \tag{95}$$

Now, the update for matrix $\boldsymbol{W}$ is needed. The derivative of L with respect to W_{ki} can be rewritten by

$$\frac{\partial L}{\partial W_{ki}} = \sum_{j=1}^{N}\frac{\partial L}{\partial h_j}\frac{\partial h_j}{\partial W_{ki}} = \frac{\partial L}{\partial h_i}\frac{\partial h_i}{\partial W_{ki}}, \tag{96}$$

where W_{ki} is the element of column k and row i in $V \times N$ matrix $\boldsymbol{W}$ and h_i again the i-th node in the projection layer. Remember than W_{ki} connects node k of the input to node i of the projection layer. Then,

$$\frac{\partial L}{\partial h_i} = \sum_{j=1}^{V}\frac{\partial L}{\partial s_j}\frac{\partial s_j}{\partial h_i} = \sum_{j=1}^{V}\varepsilon_j Z_{ij}, \tag{97}$$

as

$$\frac{\partial s_j}{\partial h_i} = \frac{\partial \boldsymbol{u}_{w_j}^T \boldsymbol{h}}{\partial h_i} = Z_{ij}, \tag{98}$$

and

$$\frac{\partial h_i}{\partial W_{ki}} = \frac{1}{C}x_{i,w_{c,l}} \quad \text{for } l = 1, \ldots, C, \tag{99}$$

which results in

$$\frac{\partial L}{\partial W_{ki}} = \frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij} x_{i,w_{c,l}} \quad \text{for } l = 1, \ldots, C, \tag{100}$$

where x_{i,w_c} is node i of input word w_c of the C context words. For all C words this results in the following SGD update for every element in matrix $\boldsymbol{W}$:

$$W_{ki}^{(new)} = W_{ki}^{(old)} - \eta\frac{\partial L}{\partial W_{ki}} = W_{ki}^{(old)} - \eta\frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij} x_{i,w_{c,l}} \quad \text{for } l = 1, \ldots, C, \tag{101}$$

where η is again the step size or learning rate. Then, every row of matrix $\boldsymbol{W}$ is updated for all C words in the context by

$$\boldsymbol{v}_{w_{c,l}}^{(new)} = \boldsymbol{v}_{w_{c,l}}^{(old)} - \eta\frac{\partial L}{\partial \boldsymbol{v}_{w_{c,l}}} = \boldsymbol{v}_{w_{c,l}}^{(old)} - \eta\frac{1}{C}\sum_{j=1}^{V}\varepsilon_j Z_{ij}\boldsymbol{x}_{w_{c,l}} \quad \text{for } l = 1, \ldots, C. \tag{102}$$

The SGD update for the document weight matrix $\boldsymbol{D}$ can be derived by taking the derivative with respect to every column p and row i of matrix $\boldsymbol{D}$. This results in the same derivation as (96), but now with respect to D_{pi}

$$\frac{\partial L}{\partial D_{pi}} = \sum_{j=1}^{N} \frac{\partial L}{\partial h_j} \frac{\partial h_j}{\partial D_{pi}} = \frac{\partial L}{\partial h_i} \frac{\partial h_i}{\partial D_{pi}}, \tag{103}$$

where the first part is derived in (97) and the second part

$$\frac{\partial h_i}{\partial D_{pi}} = b_i \tag{104}$$

which results in

$$\frac{\partial L}{\partial D_{pi}} = \sum_{j=1}^{V} \varepsilon_j Z_{ij} b_i. \tag{105}$$

This results in the SGD update:

$$D_{pi}^{(new)} = D_{pi}^{(old)} - \eta \frac{\partial L}{\partial D_{pi}} = Z_{pi}^{(old)} - \eta \sum_{l=1}^{C} \sum_{j=1}^{V} \varepsilon_j Z_{ij} b_i, \tag{106}$$

which equivalates to

$$\boldsymbol{t}_{d_I}^{(new)} = \boldsymbol{t}_{d_I}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{t}_{d_I}} = \boldsymbol{t}_{d_I}^{(old)} - \eta \sum_{j=1}^{V} \varepsilon_j Z_{ij} \boldsymbol{b}. \tag{107}$$

Distributed Bag of Words Model Updates

The projection layer in the neural network can be updated by only the linear activation matrix $\boldsymbol{D}$. With DBOW, only one document d_I is in the input layer. Therefore, the projection layer can be updated by

$$\boldsymbol{h} = \boldsymbol{D}^T \boldsymbol{b}_1 \tag{108}$$

$$= \boldsymbol{t}_{d_I}^T, \tag{109}$$

where $\boldsymbol{b}_1$ is the one-hot encoded vector belonging to input document d_I and $\boldsymbol{t}_{d_I}$ is the N-dimensional document vector from weight matrix $\boldsymbol{D}$ of document d_I. Again, column j of the weight matrix $\boldsymbol{Z}$ that connects the projection layer with the output layer is equal to vector u_{w_j}. All C output context words are randomly selected, which is the main difference between Skip-gram and DBOW, and the score for every word j in the vocabulary can be defined by

$$s_{j,l} = \boldsymbol{u}_{w_{c,l}}^T \boldsymbol{h} \tag{110}$$

$$= \boldsymbol{u}_{w_{c,l}}^T \boldsymbol{t}_{d_I}^T \quad \text{for } l = 1, \ldots, C. \tag{111}$$

Then for all C randomly selected context words, given the input document, the V-dimensional $\boldsymbol{y}_l$ vectors can be derived from

$$p(w_{c,l}|d_I) = y_{j,l} = \frac{\exp(s_{j,l})}{\sum_{i=1}^{V}\exp(s_{i,l})} \quad \text{for } l = 1,\dots,C, \tag{112}$$

where $s_{j,1} = \cdots = s_{j,C}$ which results in $y_{j,1} = \cdots = y_{j,C}$ and thus $\boldsymbol{y}_1 = \cdots = \boldsymbol{y}_C$.

Maximizing the probability that the random context output words given the input document d_I, $p(w_{c,1},\dots,w_{c,C}|d_I)$, results in the loss function to be minimized

$$L = -\log p(w_{c,1},\dots,w_{c,C}|d_I) \tag{113}$$

$$= -\log\prod_{l=1}^{C} p(w_{c,l}|d_I) \tag{114}$$

$$= -\log\prod_{l=1}^{C}\Big(\frac{\exp(s_{j^*,l})}{\sum_{i=1}^{V}\exp(s_{i,l})}\Big) \tag{115}$$

$$= -\sum_{l=1}^{C} s_{j*,l} + \sum_{l=1}^{C}\log\sum_{i=1}^{V}\exp(s_{i,l}), \tag{116}$$

where $s_{j^*,l}$ is the score of the actual context output word w_l at index j^* in $\boldsymbol{y}_l$ which is not equal for the C output context words. The loss function can be written in terms of the weight matrices $\boldsymbol{D}$ and $\boldsymbol{Z}$ depending on $\boldsymbol{u}_l$ for $l = 1,\dots,C$ by

$$L = L(s_{1,1}(\boldsymbol{D},\boldsymbol{Z}), s_{2,1}(\boldsymbol{D},\boldsymbol{Z}),\dots,s_{V-1,C}(\boldsymbol{D},\boldsymbol{Z}), s_{V,C}(\boldsymbol{D},\boldsymbol{Z})). \tag{117}$$

To update using SGD the derivative with respect to Z_{ij} one can use the multivariate function chain rule again:

$$\frac{\partial L}{\partial Z_{ij}} = \sum_{k=1}^{V}\sum_{l=1}^{C}\frac{\partial L}{\partial s_{k,l}}\frac{\partial s_{k,l}}{\partial Z_{ij}}, \tag{118}$$

where Z_{ij} connects node i of the projection layer with every index j of the output layer of the C random output context words. From (88), (89), (91) and (92), it follows that

$$\frac{\partial L}{\partial Z_{ij}} = \sum_{l=1}^{C}\frac{\partial L}{\partial s_{j,l}}\frac{\partial s_{j,l}}{\partial Z_{ij}} \tag{119}$$

$$= \sum_{l=1}^{C}(-\mathbb{1}(j = j^*)_l + y_{j,l})\frac{\partial s_{j,l}}{\partial Z_{ij}} \tag{120}$$

$$= \sum_{l=1}^{C}(-\mathbb{1}(j = j^*)_l + y_{j,l})h_i \tag{121}$$

$$= \sum_{l=1}^{C}\varepsilon_{j,l}h_i, \tag{122}$$

where $\mathbb{1}(j = j^*)_l$ is equal to 1 for context word $w_{c,l}$ if the actual context word is at index j^* of $\boldsymbol{y}_l$, and zero otherwise. The SGD update then results in:

$$Z_{ij}^{(new)} = Z_{ij}^{(old)} - \eta \frac{\partial L}{\partial Z_{ij}} = Z_{ij}^{(old)} - \eta \sum_{l=1}^{C} \varepsilon_{j,l} h_i, \tag{123}$$

which equivalates to

$$\boldsymbol{u}_{w_j}^{(new)} = \boldsymbol{u}_{w_j}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{u}_{w_j}} = \boldsymbol{u}_{w_j}^{(old)} - \eta \sum_{l=j}^{C} \varepsilon_{j,l} \boldsymbol{h} \tag{124}$$

for every word w_j in the vocabulary that is connected to column $\boldsymbol{u}_{w_j}$ in matrix $\boldsymbol{Z}$, where η is the learning rate.

The SGD update for the document weight matrix $\boldsymbol{D}$ can be derived by taking the derivative with respect to every column p and row i of matrix $\boldsymbol{D}$. The derivative of the loss function is equal to the loss function in (103), where

$$\frac{\partial L}{\partial h_i} = \sum_{j=1}^{V} \sum_{l=1}^{C} \frac{\partial L}{\partial s_{j,l}} \frac{\partial s_{j,l}}{\partial h_i} = \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij}, \tag{125}$$

and

$$\frac{\partial h_i}{\partial D_{pi}} = b_i \tag{126}$$

which results in

$$\frac{\partial L}{\partial D_{pi}} = \sum_{j=1}^{V} \varepsilon_j Z_{ij} b_i. \tag{127}$$

Then the SGD updates every element of matrix $\boldsymbol{D}$ is obtained by

$$D_{pi}^{(new)} = D_{pi}^{(old)} - \eta \frac{\partial L}{\partial D_{pi}} = D_{pi}^{(old)} - \eta \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij} b_i, \tag{128}$$

where η is the step size or learning rate. Then, every row of matrix $\boldsymbol{D}$ is updated by

$$\boldsymbol{t}_{d_I}^{(new)} = \boldsymbol{t}_{d_I}^{(old)} - \eta \frac{\partial L}{\partial \boldsymbol{t}_{d_I}} = \boldsymbol{t}_{d_I}^{(old)} - \eta \sum_{j=1}^{V} \sum_{l=1}^{C} \varepsilon_{j,l} Z_{ij} \boldsymbol{b}. \tag{129}$$

Table 7: The percentage of takeover targets in the portfolio per year and in total, and estimates of α and β for the CAPM, Fama-French Three-Factor Model and Carhart Four-Factor Model for the subset created by the top decile function for all model combinations using monthly time-series data for the returns of the stocks in the portfolio. The values for α are annualized and represent the annual excess return. α and β indicate abnormal returns for the portfolio when $t >= 1.96$ and $t <= -1.96$. No abnormal returns are created by the portfolio when $t < 1.96$ and $t > -1.96$, which is indicated by *. The different classification methods are logistic regression (LR), multinomial Naive Bayes (MNB) and the support vector machine with RBF kernel (SVM).

			Portfolio %															
Section	**Vectorization**	**Classification**	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total	α^J	β_i^J	α^{3F}	β_1^{3F}	α^{4F}	β_1^{4F}
		LR	7.3	6.3	4.5	4.8	5.1	6.4	6.0	6.9	4.5	5.7	5.25%*	1.23%	7.61%	1.03%	7.62%	1.03%
													(1.57)	(17.06)	(2.23)	(13.23)	(2.22)	(12.99)
MD&A	BoW-1	MNB	6.3	6.6	6.4	4.8	2.1	5.5	8.0	5.4	7.5	5.8	4.68%	1.29%	5.40%	1.10%	5.87%	1.08%
													(3.06)	(39.60)	(3.49)	(31.46)	(3.77)	(30.30)
		SVM	6.6	4.1	4.8	2.2	5.4	4.6	6.5	3.6	3.6	4.6	1.00*%	1.20%	3.45%	1.04%	3.75%	1.02%
													(0.65)	(35.44)	(2.20)	(28.17)	(2.37)	(27.27)
		LR	6.1	4.9	4.7	5.3	3.8	3.4	7.4	3.3	6.2	5.0	1.80%*	1.20%	4.48%	1.03%	4.78%	1.01%
													(1.25)	(38.12)	(3.05)	(30.20)	(3.23)	(29.33)
RF	BoW-1	MNB	5.8	6.1	5.3	4.3	2.9	4.9	7.9	5.4	5.9	5.4	4.67%	1.32%	5.38%	1.13%	5.73%	1.12%
													(3.01)	(39.90)	(3.42)	(31.81)	(3.63)	(30.81)
		SVM	5.8	4.6	3.8	2.5	2.9	3.7	3.2	2.1	3.3	3.5	11.12%*	1.12%	15.59%*	0.96%	14.75%*	0.99%
													(0.97)	(4.74)	(1.32)	(3.76)	(1.25)	(3.82)
		LR	8.2	7.2	7.4	3.8	5.7	6.1	4.5	4.8	5.1	5.8	2.41%*	1.22	4.67%	1.02	5.03%	1.00
													(1.46)	(33.66)	(2.77)	(26.11)	(2.97)	(25.28)
MD&A	BoW-12	MNB	7.6	6.0	6.1	3.8	2.4	5.2	7.7	5.7	6.0	5.6	5.21%	1.32	6.04%	1.12	6.62%	1.09
													(3.07)	(36.64)	(3.51)	(28.86)	(3.83)	(27.67)
		SVM	8.5	4.4	5.1	3.5	4.8	4.9	6.5	4.8	5.4	5.3	4.84%	1.27	7.84%	1.06	8.24%	1.04
													(2.04)	(24.89)	(3.23)	(19.16)	(3.37)	(18.55)

Continued on next page

Table 7 – continued from previous page

Section	Vectorization	Classification	Portfolio % 2010	2011	2012	2013	2014	2015	2016	2017	2018	Total	α^J	β_i^J	α^{3F}	β_1^{3F}	α^{4F}	β_1^{4F}
RF	BoW-12	LR	6.4	5.8	5.9	5.9	4.1	4.6	7.1	5.1	5.6	5.6	1.95% *	1.15	4.60%	0.98	4.82%	0.97
													(1.22)	(32.87)	(2.82)	(25.98)	(2.93)	(25.31)
		MNB	6.1	7.0	5.9	4.0	2.9	5.1	8.5	5.1	6.5	5.7	3.96%	1.28	4.57%	1.09	4.98%	1.07
													(2.79)	(41.99)	(3.17)	(33.28)	(3.44)	(32.09)
		SVM	5.8	6.1	5.0	5.3	3.5	4.0	8.8	6.3	5.0	5.5	2.00%*	1.30	3.64%	1.11	4.12%	1.09
													(1.21)	(36.14)	(2.18)	(28.60)	(2.45)	(27.61)
MD&A	BoW-2	LR	7.6	5.6	7.4	4.4	6.0	4.9	4.8	3.9	6.0	5.6	5.46%	1.21	7.57%	1.01	8.06%	0.99
													(2.88)	(29.55)	(3.91)	(22.88)	(4.14)	(22.03)
		MNB	8.5	6.9	6.8	4.4	3.3	4.6	7.7	6.6	5.7	6.0	7.05%	1.26	7.57%	1.06	7.79%	1.05
													(3.93)	(33.02)	(4.16)	(25.71)	(4.27)	(24.98)
		SVM	6.0	5.0	5.5	3.2	6.2	5.2	6.2	4.5	3.3	5.0	8.50%	1.24	11.48%	1.02	12.07%	1.00
													(2.72)	(18.65)	(3.58)	(14.22)	(3.74)	(13.66)
RF	BoW-2	LR	6.1	7.6	6.6	6.2	5.3	5.1	6.2	3.9	5.0	5.8	13.49% *	1.17	17.40% *	1.02	16.29% *	1.06
													(1.12)	(4.67)	(1.41)	(3.77)	(1.32)	(3.85)
		MNB	8.2	7.0	6.2	5.0	3.2	5.4	8.5	5.7	5.9	6.1	4.45%	1.26	4.90%	1.06	5.04%	1.06
													(2.87)	(37.77)	(3.12)	(29.67)	(3.20)	(28.98)
		SVM	5.8	4.9	3.8	2.2	3.2	4.0	3.2	2.1	3.3	3.6	11.37%*	1.12	15.84%*	0.95	14.97%*	0.99
													(0.99)	(4.72)	(1.32)	(3.73)	(1.27)	(3.81)
MD&A	tfidf-1	LR	10.7	9.1	8.4	6.3	6.8	5.2	11.6	6.9	6.6	7.9	4.35%	1.25%	5.14%	1.08%	5.22%	1.08%
													(2.94)	(38.82)	(3.42)	(31.22)	(3.46)	(30.59)
		MNB	5.0	2.2	6.4	2.2	4.2	3.2	4.8	3.3	3.0	3.8	1.05%*	1.14%	3.01%*	0.94%	3.19%*	0.93%
													(0.66)	(32.36)	(1.85)	(24.81)	(1.94)	(24.17)
		SVM	6.0	3.4	4.5	2.9	3.0	5.8	6.2	3.3	5.7	4.6	4.18%	1.15%	6.14%	0.98%	6.44%	0.97%
													(2.69)	(33.97)	(3.88)	(26.93)	(4.04)	(26.15)
		LR	7.9	7.9	7.8	5.3	5.6	6.3	11.8	7.2	7.4	7.5	4.41%	1.14%	5.44%	0.97%	5.47%	0.97%
													(3.57)	(42.60)	(4.34)	(33.57)	(4.34)	(32.94)

Continued on next page

Table 7 – continued from previous page

Section	Vectorization	Classification	Portfolio % 2010	2011	2012	2013	2014	2015	2016	2017	2018	Total	α^J	β_i^J	α^{3F}	β_1^{3F}	α^{4F}	β_1^{4F}
RF	tf-idf-1	MNB	5.8	5.2	6.2	5.6	3.2	3.4	4.1	3.3	3.3	4.4	3.52%	1.21%	4.79%	1.01%	5.02%	1.00%
													(2.38)	(37.51)	(3.19)	(29.13)	(3.32)	(28.39)
		SVM	4.6	5.2	5.0	5.9	5.3	3.1	4.4	5.4	4.4	4.8	6.45%*	1.34%	9.91%	1.15%	10.71%	1.12%
													(1.31)	(12.70)	(1.96)	(10.12)	(2.10)	(9.66)
MD&A	tfidf-12	LR	12.3	8.8	7.7	6.0	6.0	6.7	11.6	7.6	6.6	8.1	5.64%	1.22%	6.42%	1.04%	6.63%	1.03%
													(3.65)	(36.59)	(4.10)	(28.95)	(4.21)	(28.21)
		MNB	6.0	1.3	4.2	0.3	3.6	2.0	5.1	3.6	4.2	3.4	-2.73%*	1.08%	-0.66%*	0.92%	-0.09%*	0.89%
													(-1.88)	(33.08)	(-0.44)	(26.05)	(-0.06)	(24.86)
		SVM	10.1	5.3	6.4	3.5	3.6	7.0	6.5	5.1	5.1	5.8	6.38%	1.28%	8.41%	1.08%	8.69%	1.06%
													(2.65)	(24.77)	(3.43)	(19.30)	(3.52)	(18.77)
RF	tf-idf-12	LR	7.9	7.6	7.5	6.5	5.6	6.6	10.6	6.9	7.4	7.4	3.88%	1.15%	4.79%	0.97%	4.87%	0.97%
													(3.12)	(42.45)	(3.81)	(33.59)	(3.85)	(32.88)
		MNB	3.7	2.7	4.1	2.2	2.9	2.9	2.6	2.7	3.8	3.1	2.31%*	1.08%	4.36%*	0.88%	4.84%	0.86%
													(0.98)	(20.79)	(1.81)	(15.76)	(1.99)	(15.17)
		SVM	7.3	7.6	5.9	6.5	3.8	5.1	7.6	4.8	4.4	5.9	6.76%	1.21%	8.76%	1.04%	9.06%	1.03%
													(2.17)	(18.07)	(2.76)	(14.40)	(2.83)	(13.98)
MD&A	tfidf-2	LR	12.0	8.8	6.4	5.7	6.0	7.5	11.3	6.3	6.3	7.8	7.39%	1.28%	8.71%	1.09%	9.15%	1.07%
													(3.03)	(24.36)	(3.51)	(19.25)	(3.66)	(18.58)
		MNB	5.7	1.6	3.5	1.6	3.0	2.3	4.5	4.2	3.0	3.3	0.07%*	1.06%	2.17%*	0.91%	2.62%*	0.89%
													(0.05)	(31.01)	(1.37)	(24.66)	(1.64)	(23.72)
		SVM	8.5	4.7	5.8	2.9	4.5	5.8	6.2	4.2	4.2	5.2	7.85%	1.22%	10.57%	0.98%	11.34%	0.95%
													(2.52)	(18.37)	(3.31)	(13.72)	(3.52)	(13.04)
		LR	8.2	7.6	5.3	7.1	4.7	7.4	10.3	6.9	5.9	7.1	5.40%	1.18%	6.36%	1.00%	6.63%	0.99%
													(3.62)	(36.77)	(4.21)	(28.80)	(4.36)	(27.99)
		MNB	4.3	2.7	4.1	3.7	4.4	3.7	4.1	3.3	6.2	4.1	3.02%*	1.25%	4.55%	1.05%	5.17%	1.02%
													(1.57)	(29.96)	(2.33)	(23.27)	(2.63)	(22.38)

Continued on next page

Table 7 – continued from previous page

Section	Vectorization	Classification	Portfolio % 2010	2011	2012	2013	2014	2015	2016	2017	2018	Total	α^J	β_i^J	α^{3F}	β_1^{3F}	α^{4F}	β_1^{4F}
RF	tf-idf-2	SVM	6.4	6.1	4.7	7.8	3.2	4.0	8.5	6.3	5.3	5.8	3.53%	1.23%	5.92%	1.04%	6.19%	1.02%
													(1.96)	(31.42)	(3.22)	(24.51)	(3.35)	(23.81)
MD&A	DM-HS	LR	6.3	5.0	7.4	2.9	3.9	4.6	5.1	6.0	6.3	5.3	2.64%*	1.22%	4.36%	1.05%	4.60%	1.04%
													(1.77)	(37.36)	(2.88)	(30.05)	(3.02)	(29.22)
		SVM	5.0	2.8	3.5	2.5	2.7	2.6	4.8	3.0	3.0	3.3	8.49%*	1.25%	11.98%*	1.06%	11.12%*	1.09%
													(0.67)	(4.60)	(0.91)	(3.60)	(0.85)	(3.66)
RF	DM-HS	LR	5.5	4.6	6.6	4.3	5.6	4.3	6.5	3.3	5.3	5.1	3.54%	1.11%	5.84%	0.95%	6.31%	0.93%
													(2.31)	(33.16)	(3.74)	(26.39)	(4.01)	(25.37)
		SVM	5.5	3.0	3.8	2.5	2.6	2.6	5.0	3.0	2.7	3.4	10.19%*	1.22%	13.92%*	1.03%	13.07%*	1.06%
													(0.81)	(4.57)	(1.07)	(3.58)	(1.01)	(3.64)
MD&A	DM-NS	LR	5.4	4.7	6.8	4.4	3.9	4.9	6.5	3.6	3.6	4.9	1.89%*	1.15%	3.92%	0.97%	4.18%	0.95%
													(0.98)	(27.31)	(1.99)	(21.27)	(2.11)	(20.65)
		SVM	4.7	2.5	3.9	2.9	2.4	2.6	4.2	3.0	2.7	3.2	8.19%*	1.23%	11.57%*	1.05%	10.77%*	1.09%
													(0.64)	(4.53)	(0.89)	(3.59)	(0.82)	(3.64)
RF	DM-NS	LR	5.2	7.3	5.9	5.6	4.7	6.3	9.4	5.4	5.9	6.2	9.07%*	1.26%	10.74%	1.12%	11.12%	1.10%
													(1.69)	(11.04)	(1.96)	(9.07)	(2.02)	(8.78)
		SVM	5.5	4.6	4.7	4.3	3.2	5.1	6.5	4.5	5.0	4.8	1.18%*	1.21%	3.88%	1.01%	4.19%	1.00%
													(0.86)	(40.33)	(2.79)	(31.51)	(2.99)	(30.60)
MD&A	DBOW-HS	LR	8.2	4.1	7.1	6.7	4.8	4.6	6.0	6.9	6.9	6.1	3.94%	1.13%	5.69%	0.97%	5.84%	0.96%
													(2.73)	(36.03)	(3.87)	(28.66)	(3.95)	(27.94)
		SVM	5.4	2.8	2.9	2.9	2.7	2.6	4.5	3.0	2.1	3.2	8.80%*	1.25%	11.95%*	1.07%	10.95%*	1.12%
													(0.69)	(4.59)	(0.91)	(3.67)	(0.83)	(3.74)
RF	DBOW-HS	LR	7.6	5.5	5.3	4.7	5.9	5.4	9.1	5.1	6.2	6.1	1.98%*	1.19%	3.61%	1.02%	3.79%	1.02%
													(1.43)	(39.12)	(2.56)	(31.40)	(2.68)	(30.64)
RF	DBOW-HS	SVM	4.9	3.0	3.1	1.9	2.9	3.4	4.7	2.4	2.4	3.2	10.77%*	1.24%	14.21%*	1.06%	13.15%*	1.11%
													(0.85)	(4.66)	(1.09)	(3.69)	(1.01)	(3.78)

Continued on next page

Table 7 – continued from previous page

			Portfolio %														
Section	**Vectorization**	**Classification**	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total	α^J	β_i^J	α^{3F}	β_1^{3F}	α^{4F}	β_1^{4F}
MD&A	DBOW-NS	LR	9.5	6.0	6.1	3.5	4.8	4.6	4.5	5.7	6.0	5.6	0.64%*	1.17%	2.61%	0.99%	2.97%	0.98%
													(0.52)	(43.38)	(2.11)	(34.43)	(2.38)	(33.31)
		RBF SVM	7.9	2.5	6.1	3.5	4.5	5.2	5.7	5.4	3.3	4.9	2.69%*	1.20%	4.59%*	1.05%	5.39%*	1.02%
													(0.73)	(14.83)	(1.21)	(12.05)	(1.41)	(11.48)
RF	DBOW-NS	LR	6.1	7.3	5.9	5.0	4.1	3.7	7.4	3.3	5.3	5.3	2.71%	1.13%	5.08%	0.94%	5.55%	0.92%
													(1.59)	(30.40)	(2.92)	(23.59)	(3.17)	(22.73)
		SVM	4.3	7.0	3.4	3.4	5.6	2.6	10.0	3.6	4.4	4.9	2.95%*	1.18%	4.96%	1.00%	5.33%	0.99%
													(1.63)	(29.79)	(2.67)	(23.53)	(2.86)	(22.72)