ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

Master Thesis - MSc Econometrics and Management Science

Quantitive Finance Specialization

An Automatic Construction of a Financial Sentiment Lexicon

Author:

Tommaso GARUTTI

Student Number:

424192

Supervisor:

Flavius FRASINCAR

Second assessor:

Luuk VAN MAASAKKERS

Abstract

In this paper, we investigate the sentiment of an extensive financial corpora of 10-K and 10-Q reports. We develop two neural network formulations to automatically construct a finance-specific sentiment lexicon based on the financial reports. The first is a classification feed-forward neural network framework, while the second is regression feed-forward neural network framework. As sentiment label for a 10-K or 10-Q form, we employ the change in the company's respective stock return. In addition, we construct GARCH-based SVM and SVR models, integrated with our sentiment lexicons, to forecast stock return volatility and to evaluate the performance of the lexicons. Our lexicons were narrowly outperformed by a manually constructed sentiment lexicons and obtained better results than a recognized automatically constructed sentiment lexicon. As our lexicons are automatically built, they can be easily maintained in the future.

Date: May 1, 2021

Contents

1 Introduction

Significant advances in technology have resulted in an unprecedented amount of data being readily available online. More specifically, there has been a substantial growth in sentiment information with an explosion of short texts conveying personal opinions from social media and blogs, and extensive databases of long texts with online news sources publishing articles and official government websites releasing financial reports and documents. This has led to increased interest in sentiment analysis with particular attention from the financial research community, where investor sentiment can have important influences on fluctuations and trends on the financial market. Existing research in sentiment analysis has resulted in the development of several methods to perform sentiment mining: lexicon-based approaches, statistic or rule-based approaches, and unsupervised or supervised machine learning based approaches.

The majority of sentiment mining approaches have as the backbone of the formulation the sentiment lexicon: Feldman (2013) notes that "the sentiment lexicon is the most crucial resource for most sentiment analysis algorithms". Previous works provide evidence in support of domain-specific sentiment lexicons rather than general lexicons when analysing the sentiment of domain-specific corpora. A general sentiment lexicon would regularly assign negative weight to words such as "risk", "taxes" and "liabilities", when in a financial context these words do not necessarily carry negative sentiment (Loughran and McDonald, 2011). Loughran and McDonald compare a general semantic lexicon, the Harvard Psychosociological Dictionary, to their own finance-specific lexicon and find that three fourths of the negative words from the Harvard list are not strictly negative in a financial corpora. Therefore, the construction of an accurate domain-specific lexicon is essential for analysing the sentiment polarity and intensity of a text, especially when considering a financial corpora.

Traditionally, finance-specific lexicons have been constructed manually (Loughran and McDonald, 2011). That is, words are assigned a polarity and intensity by linguistic experts. This leads to a highly accurate sentiment lexicon, however, the manual construction of the lexicon is time consuming. In addition, as language evolves and the sentiment of words in specific fields changes, the maintenance of a static, handcrafted lexicon proves challenging.

This has led to automatic lexicon creation becoming a popular area of research in computational linguistics. This area of research can be subdivided into dictionary-based and corpus-based automatic lexicon construction. Dictionary-based approaches utilize an existing semantic lexicon or database, such as WordNet, and a set of seed sentiment words to associate to the rest

of the text. However, unless the database is domain-specific, similar misclassification issues as with general semantic lexicons may arise. Corpora-based approaches use statistical measures - such as the point-wise mutual information (PMI), that measures the associative strength of words based on frequency and vicinity to other words - to extract the sentiment lexicon from the corpus. This approach is limited by the fact that it simply takes into account frequency as a measure for association between words. More recently, machine learning based approaches have been implemented for sentiment lexicon construction. Support vector machines and especially neural networks capture latent semantic and sentiment properties by learning vector word representations by applying both linear and non-linear relations between the sentiment of the words and polarity of the entire texts. Most of the of neural network based frameworks applied on financial corpora have been focused on the analysis of short texts; generally, StockTwits with "bullish" or "bearish" labels are studied.

This increased interest in sentiment analysis with applications in finance arises an important question: how can sentiment analysis be used in a more practical financial context? Several studies have found correlations between financial measures and the sentiment of financial reports. For instance, Bonsall et al. (2017) study the relation between the readability of 10-K reports and stock price volatility, while Loughran and McDonald (2011) have linked the sentiment of 10-Ks and 10-Qs, evaluated by considering the number of positive and negatively weighted words in a document, to changes in stock return volatility after the date of publication. Despite the fact that these studies provide clear indication of a correlation between stock return volatility and the sentiment of 10-K and 10-Q reports, research regarding this topic is limited.

Modelling volatility is of critical importance to almost any participant in the financial market: an accurate volatility measure and forecast is essential to pricing assets, risk management and to making informed investment decisions. Extensive research regarding volatility modelling and forecasting results in long list of GARCH-type, introduced by Bollerslev (1986), and stochastic volatility models. Although the majority of existing volatility forecasting frameworks overlook the influence of public sentiment or sentiment of financial reports on volatility, a particular strand of literature studying this relation between sentiment and volatility is taking off. The literature, however, predominantly makes use of manually constructed lexicons to examine public opinion, by analysing sentiment of news articles, or the sentiment of 10-K and 10-Q reports, which are considered fundamental sources of company-specific information, to

then forecast volatility.

In this thesis, we aim to combine these two areas of research: first, automatically constructing a finance-specific sentiment lexicon using a machine learning based method, and, secondly, applying a GARCH-based machine learning formulation to model the relation between the sentiment of the financial corpora and stock return volatility. The financial corpora that will be analyzed in this study consists of 10-K and 10-Q reports published between 1998 and 2018, obtained from the U.S. Securities and Exchange Commission (SEC) website. The database of financial reports contains 134,000 10-Q reports and 42,000 10-K reports from 1,980 different publicly traded companies. In addition, daily stock price data, between 1998 and 2018, is collected for the 1,980 companies that will be considered in this study.

In this paper, our main contributions to the existing body of literature concerning sentiment analysis and the automatic construction of lexicons, can be described as follows. First, we investigate two different automatic approaches for constructing a finance-specific sentiment lexicon. The first is a classification neural network formulation, following the method proposed by Vo and Zhang (2016) and extended specifically for longer texts. A feed-forward neural network is applied to learn sentiment vector word representations based on the 10-K and 10-Q reports, resulting in the Classification Fin-SentiLex. As a sentiment label for the reports, we take the stock return of the corresponding company after the date of publication of the report, i.e., if the stock return after the publication date is positive, the sentiment of the report is labeled as positive, and vice-versa. The second approach used for learning a domain-specific sentiment lexicon is a regression neural network formulation. This formulation mimics the first, with one crucial difference: the sentiment of a financial report is taken as the stock return value of the corresponding company, instead of labelling the sentiment polarity as positive or negative without a magnitude. This enables the neural network based formulation to learn the sentiment lexicon considering both polarity and strength. To the best of our knowledge, our Regression Fin-SentiLex is the first automatically constructed finance-specific sentiment lexicon exploiting a numerical variable, such as the change in a company's stock return, as sentiment labels for the neural network lexicon learning process.

Secondly, we investigate the relation between the sentiment of 10-K and 10-Q forms and the corresponding company's stock return volatility by means of our GARCH-based support vector machine (SVM) and support vector regression (SVR) models. We employ two different machine learning based methods to forecast volatility; one to classify the future change in

volatility as positive or negative, and the second to forecast the stock return volatility. We formulate a GARCH-based SVM model that incorporates the sentiment of the financial reports as a feature to classify the change in volatility. Similarly, sentiment is inputted as a feature of a GARCH-based SVR model to obtain a forecast volatility. Finally, the performance of the two automatically constructed financial sentiment lexicons in classifying and forecasting volatility are compared with each other as well as to several benchmarks.

The rest of this paper is structured as follows. In Section 2 we discuss the related work on sentiment analysis and volatility modelling. Section 3 we explain our methods for constructing the finance-specific sentiment lexicon and the machine learning based methods to forecast volatility. In Section 4 we describe the benchmarks we will compare our research to and in Section 5 we further explain the data used for the study. Finally, we present our results in Section 7 and provide concluding remarks in Section 8.

2 Related Work

In this section, we first introduce literature related to sentiment lexicon learning. Following, we investigate previous literature analyzing the relation between the sentiment of financial texts and financial measures.

2.1 Lexicon Learning

Sentiment lexicon construction consists of developing a dictionary or list of words with respective sentiment polarity weights. In the domain of finance, two main methods are considered when constructing domain-specific sentiment lexicons: the manual approach and the automatic approach. The manual approach is highly labour-intensive and requires the expertise of a linguist well-informed in the specific domain of the study. Loughran and McDonald (2011) construct one of the few hand-crafted lexicons in the financial domain. They discover a great disparity between the sentiment of terms given by general sentiment lexicons and the sentiment assigned by their finance-specific lexicon. This static lexicon, however, requires a high level of maintenance and has been kept up-to-date with yearly additions and corrections to the list due to new industry-specific terms emerging and the dynamic nature of language. Automatic approaches for lexicon construction, on the other hand, do not need a linguistic expert, but require a well-formulated method for learning syntactic or statistical properties to extract sentiment

from a domain-specific corpus. The class of automatic approaches to build sentiment word lists can be further subdivided into three categories: the dictionary-based approach, the statistics- or rule-based approach, and the machine learning based approach.

Dictionary-based approaches generally exploit and augment an existing dictionary, or a database of words, employing a set of statistical or, more commonly, syntactic rules to build the semantic lexicon. That is, a small set of seed words is chosen as an initial representation for each category - i.e., positive and negative sentiment - and syntactic rules are applied to learn the polarities of other terms. Hatzivassiloglou and McKeown (1997) are amongst the first to use syntactic rules to determine the polarity of terms. Their method initially defines a set of seed words and analyzes pairs of adjectives conjoined by linguistic conventions, such as "and" or "but", to determine the sentiment polarity of terms. This research is limited to determining the orientation of adjectives only. Esuli and Sebastiani (2005) propose a method employing the large-scale WordNet[1] database. Synsets[2] are exploited by Esuli and Sebastiani to learn the sentimental orientation of new terms based on a small set of seed words for positive and negative categories. They extend their own method (Esuli and Sebastiani, 2006b; Esuli and Sebastiani, 2006a) resulting in the semi-supervised sentiment learning resource SentiWordNet 3.0 (Baccianella et al., 2010). While generally syntactic rules are utilized to build the sentiment lexicon upon the chosen set of seed words, several different rule-based approaches, for extending the seed set of words, have been studied. Turney and Littman (2003) implement pointwise mutual information (PMI), a statistical measure of semantic association between words or phrases, in order to construct a sentiment lexicon: the frequency of co-occurrence between a new term and terms from the seed set can be used to determine the polarity and strength of the new term. On the other, Rao and Ravichandran (2009) combine the syntactic relations defined in WordNet and graph theory to map a document or sets of documents to graphs. They construct a semi-supervised graph-based sentiment lexicon learning method, where each graph consists of nodes, each representing a term, and edges between nodes, where the edge defines a relationship between the terms and the weight associated with the edge represents the strength and sign of the relationship: positive implies the terms are synonyms or have similar connotations, and negative implies they are antonyms or opposing connotations. Kamps et al. (2004) bridge these two contrasting ideas and construct a statistical measure to estimate the shortest distance between nodes (or terms); this value represents the strength of the relationship between two terms.

[1] An extensive lexical database where words are linked by semantic relations, such as synonyms or antonyms.
[2] A set of words linked by lexical relations (i.e., synonyms).

The statistic-based approaches implement numerical measures or instruments to extract sentiment polarities from the domain-specific corpus. The employment of statistical measures for sentiment lexicon construction can be approached from two different directions: basing the research on the orientation of a small set of seed words and extending the lexicon using statistical measures (Turney and Littman, 2003), or automatically constructing the sentiment lexicon by applying the numerical measures to a domain-specific corpus with sentiment labels. Mohammad et al. (2013) formulate an automatic supervised learning method that exploits the PMI measure to learn term polarity for a labeled corpus of tweets, where emoticons in tweets are adopted as positive or negative sentiment labels. That is, the PMI value is calculated between a term and the label of the corresponding text and utilized to learn the lexicon. One of the few finance-specific sentiment lexicons learned exploiting statistic measures is developed by Oliveira et al. (2014): Information Gain (IG), TF-IDF, Class Percentage (CP) and Weighted Class Probability (WCP) are amongst the numerical instruments employed in this formulation to extract a finance-specific sentiment lexicon from StockTwits messages with "bullish" or "bearish" orientation labels.

In recent years, a growing body of literature in the field of sentiment analysis began utilizing machine learning based methods for the construction of domain-specific sentiment lexicons. The majority of deep learning methods implement a supervised learning framework applied on a large corpora of short texts. A common method used for sentiment classification is distant supervision (Davidov et al., 2010; Kouloumpis et al., 2011; Pak and Paroubek, 2010; Tang et al., 2014b; Vo and Zhang, 2015, 2016), where short texts have corresponding labels representing sentiment; in this case, tweets and emoticons. Similarly, for a corpora of finance-specific short texts, such as StockTwits, financial blogs and news headlines, "bullish" or "bearish" labels represent positive and negative sentiment (Li and Shah, 2017; Zhang et al., 2018). In order to precisely learn sentiment features from distant-supervised texts using machine learning techniques, an accurate vector-space representation of terms is necessary.

Research in word embedding algorithms has been primarily focused on syntactic-based algorithms (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014; Socher et al., 2012), however, for the purpose of sentiment analysis, Tang et al. (2014b) demonstrates that syntactic algorithms are not effective since vector word representations are task-specifc. Tang et al. (2014b) propose a method for extracting continuous word representation for sentiment classification from a corpus of labelled tweets. The feed-forward neural

network learning algorithm based on syntactic properties developed by Collobert et al. (2011) is extended to include sentiment supervision to learn sentiment-specific word embeddings from distant-supervised tweets. Further complementary research by Tang et al. (2014a) is carried out, where sentiment learning is considered as a phrase-level classification. A neural network based formulation is constructed integrating both syntactic and sentiment properties by means of a weighted linear combination of an unsupervised syntactic learning model and a supervised sentiment-specific model. The neural network model is trained and applied to extend an existing dictionary to construct a sentiment-specific phrase-level lexicon. Astudillo et al. (2015) and Amir et al. (2017) investigate task-specific word representation by means of a Non-Linear Subspace Embedding model (NLSE). The NLSE is defined as a feed-forward neural network, and is exploited to expand lexicons constructed by unsupervised word embeddings to task-specific adaptations. Vo and Zhang (2016) construct a simple neural network formulation, with a single hidden layer, to automatically learn sentiment lexicon based on distant-supervised tweets written in several languages. The vector word representation, used in this framework, is a 2-dimensional vector representing positive and negative polarities in a sentiment-specific application. In previous work, Vo and Zhang (2015), investigated the combination of two different word embedding techniques on a large data set of manually labeled tweets. The combination of the *word2vec* method, developed by Mikolov et al. (2013), and the Sentiment-Specific Word Embedding model, formulated by Tang et al. (2014b), is employed to extract a set of more accurate vector word representations. The word embeddings are then utilized in sentiment classification, by making use of a feature extraction technique.

The existing literature in machine learning based models for finance-specific lexicon creation is limited. Zhang et al. (2018) construct an automatic supervised financial text mining report on short texts from StockTwits. The Context-Aware Deep Embedding Network (CADEN) is an adaptive neural network formulation that creatively combines user embeddings with syntactic and semantic properties. Li and Shah (2017), extend the sentiment-specific word embedding procedure to a financial corpora. Similarly to Vo and Zhang (2016) and Esuli and Sebastiani (2006a), they utilize a 2-dimensional vector representation for each term for sentiment-specific embedding. The feed-forward neural network, inspired by Tang et al. (2014b) and Collobert et al. (2011), is then utilized to construct a finance-specific sentiment lexicon without any hand-crafted features.

In our study, we take inspiration from Esuli and Sebastiani (2006a), Li and Shah (2017),

and Vo and Zhang (2016) and employ a 2-dimensional vector representation for each term. The feed-forward neural network, used to learn the word vector representations, is structured such that the two attributes of the vector represent the positive and negative values of each term. While simple, this vector representation allows for easy interpretation and performs remarkably well for sentiment classification.

2.2 Financial Measures and Sentiment Analysis

Sentiment analysis applied to finance has been a growing research topic due to the interest in understanding public opinion; that is, quantifying the sentiment of news articles, tweets and financial reports. Tetlock (2007) uses quantitative measures to translate media opinions of the stock market, using data from the Wall Street Journal, into quantitative data. He discovers that negative sentiment especially can negatively pressure the market, while high levels of positive and negative sentiment influences market trading volumes. Similar sentiment studies result in clear correlations between sentiment and financial market measures. Loughran and McDonald (2011) link the frequency of positive and negative terms in a financial 10-K report to stock return volatility, and Bonsall et al. (2017) associate their measure for the readability of a financial report to volatility. Zhang and Skiena (2010) use a previously constructed general sentiment lexicon to analyse the sentiment of blog and news sources, then designing a trading strategy, to which they feed the sentiment information, that leads to favorable results. Oliveira et al. (2017) study tweets data and the influence of public opinion on the stock market. They analyse the sentiment of tweets by means of statistical measures (Oliveira et al., 2014) and find that tweet sentiment can be useful in the prediction of returns, particularly of the S&P 500 index. The relation between stock return volatility and the sentiment of financial reports is investigated by Wang et al. (2013). The sentiment of the reports is extracted by employing the finance-specific sentiment lexicon manually constructed by Loughran and McDonald (2011), and a support vector regression (SVR) model is used to model the relation between volatility and sentiment.

To this end, we will similarly investigate the relation between the volatility of stock returns and the sentiment of financial 10-K and 10-Q reports. However, we construct the sentiment lexicon ourselves by means of two neural network formulations, a binary classification model and a continuous model, and evaluate the influence of the sentiment on stock return volatility by employing a GARCH-based SVM and SVR. The methodology is further explained in the following section.

3 Methodology

In this study, we formulate a two-step problem to model the relation between the sentiment of the corporate reports, 10-K and 10-Q reports, and the company's future financial risk. First, an automated approach is applied to construct a finance-specific sentiment lexicon based on the 10-K and 10-Q reports. We formalize a feed-forward neural network based approach to obtain a dictionary of words with corresponding positive or negative sentiment scores. The second step consists of mapping the relation between a document's sentiment score and the future fluctuations in stock return volatility.

In this section, we describe the methods used to perform this two-step formulation. In Section 3.1, we briefly introduce the volatility measure used for the study, Section 3.2 describes the neural network based approach for automatic sentiment lexicon construction, and lastly, in Section 3.3, the relation between the sentiment of the corporate reports and the volatility is modelled.

3.1 Stock Return Volatility

Stock return volatility is a measure indicating the variation of a stock's return from its mean. In this study, we follow the definition from Wang et al. (2013) for stock return volatility from time $t-n$ to t, which is given by

$$\sigma_{[t-n,t]} = \sqrt{\frac{\sum_{i=t-n}^{t}(R_{[i-1,i]} - \bar{R})^2}{n}}, \quad \bar{R} = \frac{\sum_{i=t-n}^{t} R_{[i-1,i]}}{n+1}, \tag{1}$$

where the stock's return $R_{[t-n,t]}$ is defined as holding the stock for n periods of time (Tsay, 2005), as follows

$$R_{[t-n,t]} = \frac{S_t - S_{t-n}}{S_{t-n}}, \tag{2}$$

where S_t is the stock price at time t. In this study our textual data of 10-K and 10-Q forms is quarterly data. In parallel, we take n as one quarter, therefore making stock return and stock return volatility quarterly variables. For the rest of this paper, we denote $R_t := R_{[t-n,t]}$ and $\sigma_t := \sigma_{[t-n,t]}$ with n as one quarter.

3.2 Sentiment Lexicon Construction

For sentiment analysis, it is crucial to accurately convert qualitative data into quantitative data. To address this problem, we attempt to formulate precise supervised automatic approaches to extract sentiment by assigning polarity values to words based on the financial corpora of 10-K and 10-Q forms. Corporate reports, however, do not have natural labels, such as emoticons for tweets and "bullish" or "bearish" labels for StockTwits. Therefore we turn to the stock market for a proxy of sentiment: we take an increase in the stock's return R_t over the quarter following the filing of the report as an indicator of positive sentiment, and a decrease as negative sentiment. We develop two different neural network based formulations in order to automatically construct the finance-specific sentiment lexicon: the first being a classification feed-forward neural network and the second a regression neural network.

3.2.1 Classification Neural Network

In Figure 1, we illustrate the architecture of our classification neural network: a feed-forward neural network implemented for learning the sentiment of words or phrases by predicting the sentiment probability of the financial reports. We follow Li and Shah (2017) and Vo and Zhang (2016) for the vector representation of words and use a two-dimensional vector to quantitatively define the sentiment of a word. That is, a term (word or phrase) is defined as $w_i = (p_i, n_i)$, where p_i and n_i are the positive and negative polarity values of the term. This simple vector representation allows for a straightforward identification of the polarity of the term, and to discern whether the term has a significant sentiment. We assign positive and negative sentiment labels to each 10-K or 10-Q form based on the change in stock return following the publication of the respective document. That is, for each document d_k of our corpora of financial texts, we define an indicator variable r_k, that takes value 1 if the stock return R_t increases after the filing date of the report, and 0 if if R_t decreases. The indicator r_k is given as

$$r_k = \begin{cases} 1 & \text{if } R_{t+1} \geq R_t \\ 0 & \text{if } R_{t+1} < R_t \end{cases},$$

where R_t is the stock return at the filing date and R_{t+1} is the return one quarter after. To adhere to the structure of our classification neural network, we define document sentiment as a two-dimensional vector: [1,0] if $r_k = 1$ (positive sentiment), and [0,1] if $r_k = 0$ (negative sentiment).

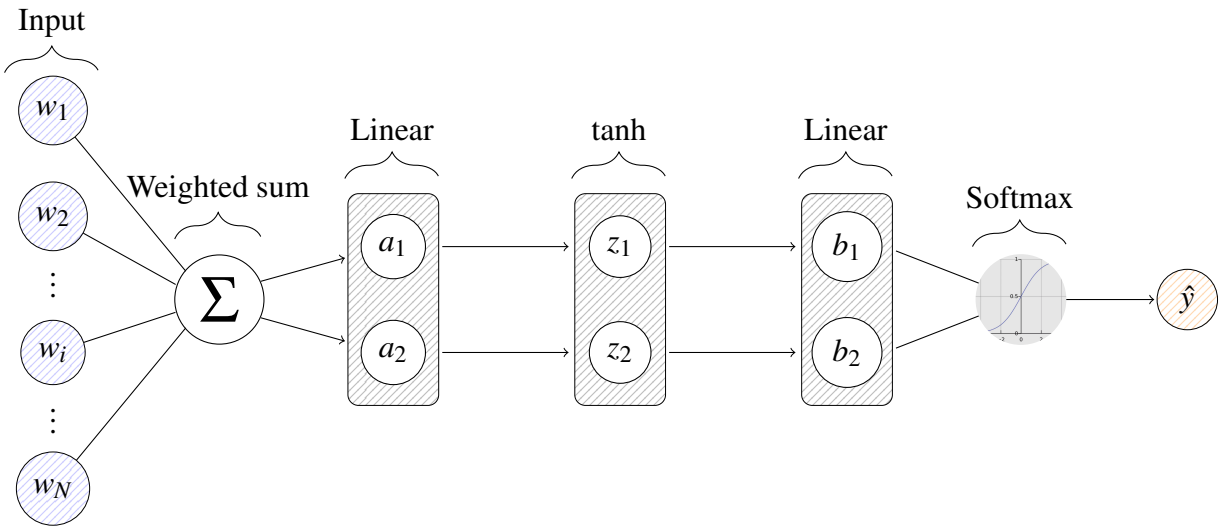

Figure 1: Classification-based neural network model.

The first layer in the neural network is the input layer. For every document d_k, a sequence of words $w_1, w_2, ..., w_N$ is extracted. Since texts vary in length, a common method employed to input an entire text into a neural network is using ngrams, or a sequence of n words or tokens, as input and slide it across the length of the entire text (Tang et al., 2014b). While this is reasonable for short texts, in our case, this would be a very expensive task as the size of 10-K and 10-Q reports can exceed hundreds of pages. Vo and Zhang (2016) sum over all the words in a text, such that a two-dimensional vector, containing the sum of all positive and all negative scores, is passed on to the next linear layer. For our neural network implementation, we adapt Vo and Zhang's idea of a summation layer for long texts, such as the 10-K and 10-Q forms. Since the financial reports are consisted of, on average, 30,000 words, a simple summation layer would result in frequent words having a much greater impact on the document sentiment than rare words. However, a word's frequency does not necessarily translate to significance or sentiment weight. Therefore, we construct a weight $\omega_{i,k}$ for each word i in document d_k based on the frequency of the word in document d_k, $f_{i,k}$, the average word frequency in document d_k, $\bar{f}_k$, and the inverse document frequency statistic:

$$\omega_{i,k} = \begin{cases} \frac{1+log(f_{i,k})}{1+log(\bar{f}_k)} log\frac{K}{df_i} & \text{if } f_{i,k} \geq 1 \\ 0 & otherwise \end{cases}, \tag{3}$$

where K is the total number of documents and df_i is the document frequency of word i, that is, the number of documents word i appears at least once in.

The input layer of our neural network formulation takes the sum of all words in a specific

document d_k and multiplies it by the corresponding weight $\omega_{i,k}$. The input layer is, therefore, defined as

$$h = \sum_{i=1}^{N} \omega_{i,k} w_i, \tag{4}$$

for document d_k, where w_i is the embedding of word i.

The first linear layer is illustrated in Equation (5), where the resulting vector of sums h is transposed and multiplied by a diagonal weight matrix D:

$$a = Dh^{\intercal}, \quad D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}. \tag{5}$$

The resulting vector therefore has dimensions 2×1. The expanded layer is given by,

$$a = \left[D_1(\textstyle\sum_{i=1}^{N} \omega_{i,k} p_i), \quad D_2(\textstyle\sum_{i=1}^{N} \omega_{i,k} n_i) \right]^{\intercal}, \tag{6}$$

where p_i and n_i are the positive and negative scores of word i in section s. Following, a hyperbolic tangent is incorporated in the neural network framework to capture the non-linear relation:

$$z = tanh(a) = \frac{e^{2a} - 1}{e^{2a} + 1}. \tag{7}$$

The second linear layer is applied as shown below,

$$b = Wz + c, \tag{8}$$

where W is fixed as a diagonal matrix and c is an additional bias term. We fix both the W matrix in Equation (8) and the D matrix in Equation (5) to diagonal in order to keep the positive and negative word scores separate. Finally, a softmax function is implemented as the final activation layer to evaluate the sentiment probability $\hat{y}$ of the document:

$$\hat{y} = softmax(b). \tag{9}$$

The softmax function is defined below,

$$softmax(b_n) = \frac{e^{b_n}}{\sum_{j=1}^{2} e^{b_j}}. \tag{10}$$

Therefore, the probability distribution for a two class problem, either positive or negative sentiment, is given by,

$$\hat{y} = \Big[f_\theta(w), \quad 1 - f_\theta(w) \Big]^\intercal. \tag{11}$$

Expanding using the definition of the softmax function in Equation (10) and the layers described above, the probability is evaluated as,

$$\begin{aligned} f_\theta(w) &= \frac{e^{b_1}}{e^{b_1} + e^{b_2}} = \frac{1}{1 + e^{b_2 - b_1}} \\ &= \frac{1}{1 + e^{W_2(tanh(D_2(\sum_{i=1}^{N} \omega_{i,k} n_i)) + c_2 - W_1(tanh((D_1(\sum_{i=1}^{N} \omega_{i,k} p_i)) + c_1}}. \end{aligned} \tag{12}$$

This parallel structure, for positive and negative scores, is formulated such that the positive scores positively influence $f_\theta(w)$ and negatively influence $1 - f_\theta(w)$, and vice versa for the negative scores. The probability $f_\theta(w)$ is the probability that the sentiment of the document is positive, while $1 - f_\theta(w)$ is the probability that the sentiment is negative. The cross-entropy loss function is used as the objective function:

$$loss = -\sum_{k=1}^{K} y_k log(\hat{y}_k) = -\sum_{k=1}^{K} loss_k, \tag{13}$$

where y_k is the true sentiment of document d_k: if the stock return R_t increases after the publication of document d_k, y_k = [1,0] (positive class); if R_t decreases, y_k = [0,1] (negative class). Since this is a two-class classification problem, the loss function can be simplified to

$$loss_k = \begin{cases} log(f_\theta(w)), & if \;\; y_k = [1,0]^\intercal \\ log(1 - f_\theta(w)), & if \;\; y_k = [0,1]^\intercal \end{cases}. \tag{14}$$

3.2.2 Regression Neural Network

Figure 2 illustrates the architecture of the regression neural network. We follow a very similar structure to the classification neural network defined in Section 3.2.1. The same two-dimensional vector representation is used for each word $w_i = (p_i, n_i)$. The main distinction is the output of the neural network. In this case, instead of having a simple classification problem, we take y_k as the percentage increase or decrease in stock returns R_t.

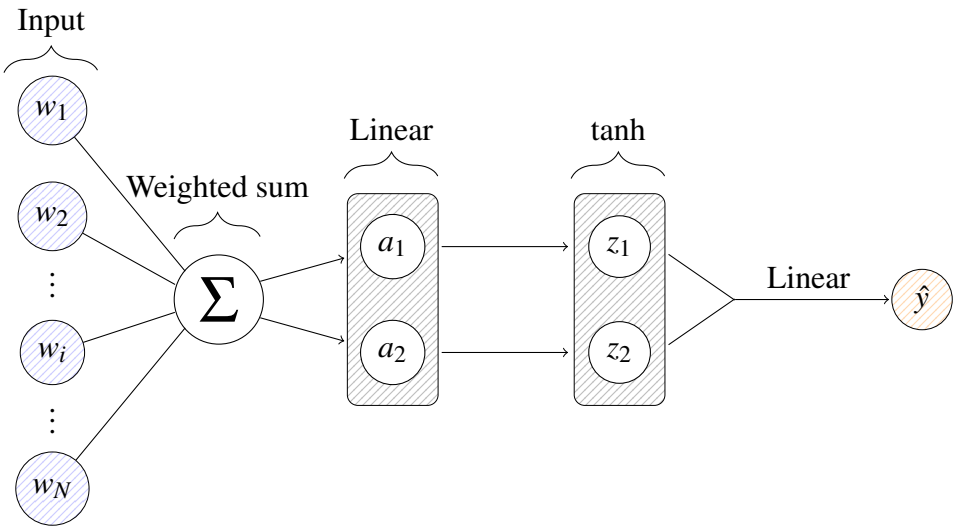

Figure 2: Regression-based neural network model.

We apply the same weighted summation procedure as described in Equation (4) to sum over the vector representation of the terms. The first linear layer and the non-linear hyperbolic tangent layer also mimic those described in Equations (5) and (7), respectively. However, since the output of the model is a numerical value and not a class, the final layer of the model is formulated as a linear layer:

$$\hat{y} = \delta z, \tag{15}$$

where z is the output of the non-linear layer (Equation (7)) and δ is a 1×2 vector.

Finally, the mean squared error (MSE) loss function is employed as the loss function:

$$loss = \frac{1}{K} \sum_{k=1}^{K} (y_k - \hat{y}_k)^2. \tag{16}$$

3.3 Modelling Volatility

From Section 3.2 we obtain two different finance-specific sentiment lexicons: the first is constructed by means of a classification neural network (Classification Fin-SentiLex) and the second by a regression neural network (Regression Fin-SentiLex). Given the sentiment lexicon, the next step is to model the relation between the sentiment of the financial reports and the volatility of the respective stock return. In order to model this relation, we use support vector machines, which is a commonly used machine learning technique in sentiment analysis (Vo and Zhang, 2016; Li and Shah, 2017; Wang et al., 2013). Two different supervised learning techniques are used: support vector machine (SVM) and support vector regression (SVR), where the SVM is

used for classification and the SVR for regression. We develop a GARCH adaptation of both machine learning techniques.

The GARCH model, introduced by Bollerslev (1986), is an autoregressive volatility model. We implement a simple adaptation of the GARCH(p,q) model as defined below,

$$R_t = \mu_t + \varepsilon_t, \quad \varepsilon_t | \psi_{t-1} \sim N(0, \sigma_t^2)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \sigma_{t-i}^2 + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j}^2, \quad (\alpha_0 > 0; \alpha_i, \beta_j \geq 0),$$

where R_t are the stock returns at time t, μ_t is the mean of the R_t over the past year (i.e. the past four quarters), ψ_{t-1} is the information known at time $t-1$, σ_t^2 is the corresponding volatility at time t, and ε_t is the shock term.

Both SVMs and SVRs are supervised learning methods that utilize a set of input features and output values to construct a hyperplane that separates the data. In our case, volatility is the output value of the SVM and SVR. For the classification SVM, we construct trend volatility variable v_k taking values 1, if the stock return volatility after publication of document d_k increases, and 0 if the volatility decreases (or remains constant). That is,

$$v_k = \begin{cases} 1 & \text{if } \sigma_{t+1}^2 \geq \sigma_t^2 \\ 0 & \text{if } \sigma_{t+1}^2 < \sigma_t^2 \end{cases},$$

where σ_t^2 is the stock price volatility at the time of filing document d_k, and σ_{t+1}^2 is the volatility one quarter after. For the regression model (SVR) we use σ_k^2, the volatility of the stock return after publication of document d_k (i.e. σ_{t+1}^2 corresponding to document d_k), as output. In order to incorporate GARCH-based features in the two machine learning models, we construct the input features matrix $\mathbf{x}_k$ such that it includes a past volatility terms σ_k^2 and the shock terms ε_k^2, consisting of p and q lagged terms respectively, in addition to the sentiment terms Ω_k of document d_k, as shown in Figure 3.

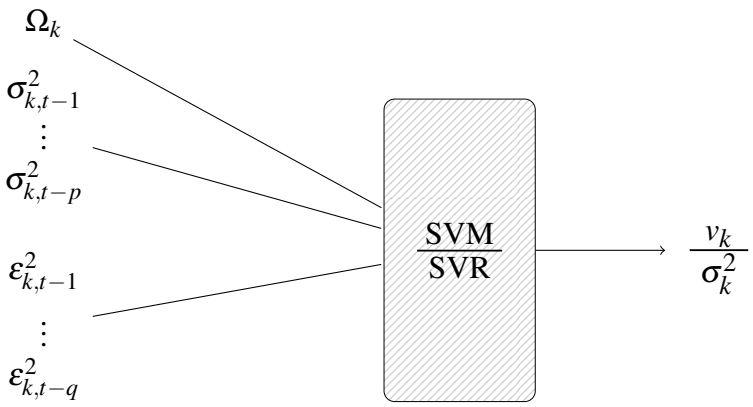

Figure 3: SVM and SVR models, with the input features and corresponding outputs.

The lexicon feature matrix, Ω_k, includes four sentiment-based features: the score of each document d_k, obtained by summing the negatively and positively orientated terms in document; the ratio of positive to negative terms; the number of negative terms; and a measure for readability of the document, which we take as the length (word count) of the 10-K or 10-Q form.

In addition to forecasting the volatility indicator v_k and the actual volatility σ^2_k, we mimic this SVM model to forecast the stock return indicator, r_k, used as sentiment label in the lexicon learning procedure. That is, we employing the same SVM structure, but using the lexicon feature matrix, Ω_k, as only input feature.

3.3.1 Support Vector Machine

Support Vector Machines (SVMs), developed by Boser et al. (1992), are supervised learning algorithms that use linear or non-linear rules to predict the class of a variable based on the input data. Specifically, SVMs create a set of hyperplanes in an high-dimensional space to accurately classify the data points. In this case, we have a binary classification problem: whether the change in volatility in the quarter after the release of the financial report is positive or negative.

Our training dataset consists of $(\mathbf{x}, v)$, where $\mathbf{x}$ is the matrix of features (input of the SVM) and v is the class (output of the SVM). In our case, the input matrix $\mathbf{x}_k$ consists of GARCH-based features, σ^2_k and ε^2_k, and sentiment features Ω_k for document d_k, and the classes are defined by v_k. A binary SVM with a linear hyperplane and soft margins, applied over a set of documents $\{d_1, d_2, ..., d_k, ..., d_N\}$, can be defined as,

$$v_k = \begin{cases} 1 & if \quad \mathbf{w}^\intercal \mathbf{x}_k + b \geq 1 \\ 0 & if \quad \mathbf{w}^\intercal \mathbf{x}_k + b \leq 1 \end{cases},$$

with primal optimization problem:

$$\begin{aligned} &\underset{\mathbf{w}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + C\sum_{k=1}^{N}\xi_k \\ &\text{subject to:} \begin{cases} v_k(\mathbf{w}^\intercal\mathbf{x}_k + b) \geq 1 - \xi_k, \\ \xi_k \geq 0, \end{cases} \quad for\ i = 1,...,n. \end{aligned}$$

where $\mathbf{w}$ and b are the parameters to be optimized, and ξ_k are the slack variables. Using the Lagrangian, the optimality condition $\mathbf{w} = \sum_k^n \alpha_k v_k \mathbf{x}_k$ is found. Hence, the above primal optimization problem can be transformed into the dual problem:

$$\begin{aligned} &\underset{\alpha}{\text{maximize}} \quad \sum_{k}^{n}\alpha_k - \frac{1}{2}\sum_{k}^{N}\sum_{j}^{N}\alpha_k\alpha_j v_k v_j \mathbf{x}_k^\intercal\mathbf{x}_j \\ &\text{subject to: } 0 \leq \alpha_k \leq C, \quad for\ i = 1,...,n. \end{aligned}$$

In order to perform non-linear classification tasks, the inner product $\mathbf{x}_k^\intercal\mathbf{x}_j$ can be replaced with kernels $K(\mathbf{x}_k, \mathbf{x}_j) = \varphi(\mathbf{x}_k)^\intercal\varphi(\mathbf{x}_j)$ that map the feature space onto a higher-dimensional feature space.

3.3.2 Support Vector Regression

Support Vector Regression Machines (SVRs) are an extension of SVMs introduced by Drucker et al. (1996). SVMs maximize the distance between the separating hyperplanes and the decision boundaries of the classes. The regression extension uses the same methods as SVM to fit a linear or non-linear function to a set of data with at most a deviation of ε from the target output v. In our case, we attempt to fit a non-linear function $f(\mathbf{x}_k, \mathbf{w})$ mapping the relation between the stock return volatility σ_k^2 to a set of features $\mathbf{x}_k$,

$$v_k = f(\mathbf{x}_k, \mathbf{w}) + b,$$

with the following optimization problem,

$$\underset{\mathbf{w}}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + C\sum_{k=1}^{N}(\xi_k - \xi_k^*)$$

$$\text{subject to:} \begin{cases} v_k - (\mathbf{w}^\intercal\varphi(\mathbf{x}_k)) - b \leq \varepsilon + \xi_k, \\ (\mathbf{w}^\intercal\varphi(\mathbf{x}_k)) + b - v_k \leq \varepsilon + \xi_k^*, \quad for\ i = 1,...,n. \\ \xi_k, \xi_k^* \geq 0, \end{cases}$$

The constraints of the primal problem displayed above allow for the error region ε, meaning that the errors within the ε-region do not contribute to the cost: the prediction is, therefore, less sensitive to small errors. The dual optimization problem is given by,

$$\underset{\alpha}{\text{maximize}} \quad -\frac{1}{2}\sum_{k}^{N}\sum_{j}^{N}(\alpha_k - \alpha_k^*)(\alpha_j - \alpha_j^*)(\varphi(\mathbf{x}_k)^\intercal\varphi(\mathbf{x}_j)) - \varepsilon\sum_{k}^{N}(\alpha_k + \alpha_k^*) + \sum_{k}^{N} v_k(\alpha_k + \alpha_k^*)$$

$$\text{subject to:} \begin{cases} \sum_k^N(\alpha_k - \alpha_k^*) = 0 \\ 0 \leq \alpha_k, \alpha_k^* \leq C, \end{cases} \quad for\ i = 1,...,n,$$

where $\varphi(\mathbf{x}_k)^\intercal\varphi(\mathbf{x}_j)$ is the kernel function used to map the feature space into a higher dimension.

4 Benchmarks

To evaluate the accuracy of the sentiment lexicons constructed by means of our Classification Neural Network and Regression Neural Network, we compare the performance of the volatility forecasts relative to benchmark sentiment lexicons. In this section, we describe the two benchmark sentiment lexicons used in our study to evaluate the performance of our lexicons.

4.1 Loughran and McDonald

The first benchmark lexicon considered in our study is Loughran and McDonald's manually constructed sentiment lexicon. The word list was developed by studying over 50,000 10-K documents filed between 1994 and 2008 and analysing all words with a minimum frequency of 5% in the financial documents. The complete word list consists of seven sentiment-specific word lists: financial-positive, financial-negative, financial-uncertainty, financial-litigious, strong modal words and weak modal words. However, in order to remain aligned with the structure of our sentiment lexicons and accurately evaluate the performance of the forecasts, we only considered

the financial-positive (Fin-Pos) and financial-negative (Fin-Neg) word lists. The two word lists consist of 353 and 2,337 words, respectively.

4.2 Vo and Zhang

The second benchmark lexicon, is a sentiment lexicon constructed by means of a machine learning based approach. Vo and Zhang (2016) employ a simple but efficient neural network method to automatically learn a sentiment lexicon. We take inspiration from this paper, for our classification neural network, with the simple summation procedure and the use of a softmax output function.

Vo and Zhang's neural network model implements a 2-dimensional vector representation for each word, such that each word is represented as $w_i = [p_i, n_i]$, where p_i represents the positive value and n_i the negative value of the word i. The feed-forward neural network has a simple structure. The input layer of the neural network takes the sum of the vector representation of each word in a document,

$$h = \sum_{i=1}^{N} w_i.$$

The sentiment probabilities are evaluated using a softmax function as follows,

$$\hat{y} = Softmax(hW),$$

where W is a diagonal coefficient matrix. The sentiment lexicon is automatically learnt by applying backpropagation over a cross-entropy loss function to optimize the positive and negative values (p, n) for each word.

5 Data

In this section, the textual and numeric dataset used to learn and test our automatically constructed sentiment lexicons, are introduced. Additionally, the parsing procedure for the financial reports is explained.

5.1 Text Dataset

For our analysis, we use an extensive dataset of 10-K and 10-Q financial reports retrieved from the SEC database through the EDGAR[3] file search and access system. 10-K and 10-Q reports are yearly and quarterly documents, respectively, containing company-specific information, such as, a company's financial statement and a discussion of the main risks a company faces. Generally, the yearly 10-K form is more comprehensive than the 10-Q form, however both contain crucial and timely information regarding the company's performance. Any publicly traded company is legally required by the Securities and Exchange Commissions (SEC) to submit the financial reports within a specified time-frame. Laws and regulations, set by the SEC, require companies to disclose all information regarding the financial performance of the company; that is, any false or misleading statements, as well as omitted information, are strictly prohibited by the SEC. Therefore, a company's 10-K and 10-Q reports provide a detailed and accurate overview of a company's business and the risks it may face, allowing investors to fully understand the financial proceedings of a company.

A financial 10-K or 10-Q report is structured into four parts and fifteen items, as displayed in Figure 4. Analysts and investors are generally interested in part 2. Specifically, tables and figures in Item 6, Consolidated Financial Data, and Item 8, Financial Statements, are analyzed to grasp an understanding of a company's financial state. In our study, we focus on the textual aspect of the financial reports instead of the financial figures. According to the SEC 10-K form guidelines[4], the items that contain significant textual information are Item 1A, Item 7 and Item 7A. Item 1A, Risk Factors: includes a discussion of the relevant risk factors threatening a company. Item 7, Management's Discussion and Analysis of Financial Condition and Results of Operations: contains a discussion of a company's financial performance, financial transactions, accounting changes or adjustments and analysis of financial risks the company faces. Item 7A, Quantitative and Qualitative Disclosures about Market Risk: provides information regarding a company's exposure to market risk.

[3] https://www.sec.gov/edgar/search-and-access

[4] https://www.sec.gov/fast-answers/answersreada10khtm.html

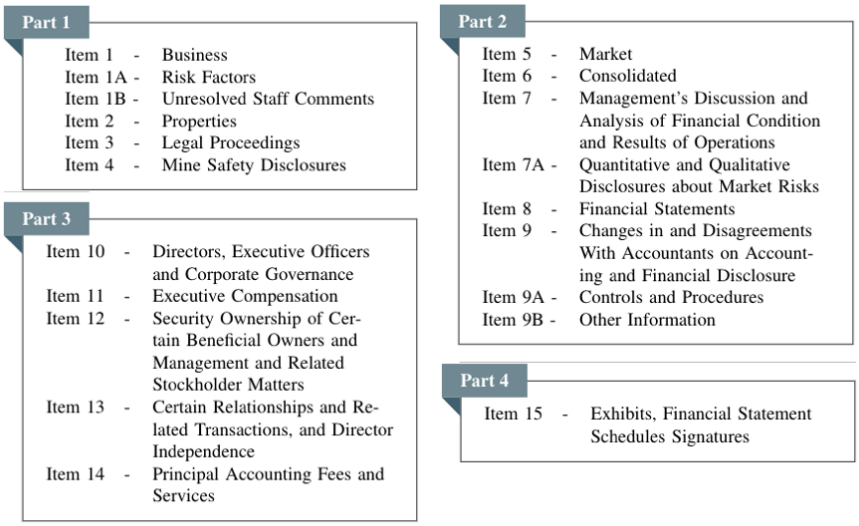

Figure 4: Structure of a 10-K form.

We construct our dataset of 10-K and 10-Q forms by matching a company's Central Index Key (CIK) with the financial reports in the EDGAR system and extracting the forms. The unfiltered dataset consists of 867,742 financial reports, of which 249,470 are 10-K forms and 618,272 are 10-Qs. This dataset is composed of financial report from 38,808 different companies between 1994 and 2018.

The raw text dataset is subsequently filtered according to several requirements we impose in order to carry out our analysis. First, we require that the difference between the 'Filed as of Date' and 'Conformed Period of Report' of all 10-K or 10-Q reports is not greater than 6 months. The 'Conformed Period of Report' date is the end date of the span time covered by the financial report, while the 'Filed as of Date' date is the official filing acceptance date of the SEC. If a form's filing date is significantly distant from the report period, it is likely that the information in the report has been leaked or been included in other financial reports from the company. Therefore, the change in price, following the release of the 10-K or 10-Q form, may not be directly linked to the sentiment of the financial report. Since our research is centered around the effect of the sentiment of the 10-K and 10-Q forms on the change in stock prices and, consecutively, the stock return volatilities, we remove any financial report who's filing date is more than 6 months distant from the period of the report.

Secondly, we filter out any financial form with a word count smaller than 2000 words. We require that the form is long enough to convey tone and sentiment in the significant sections, such as items 7 and 7A. In some cases, sections are referred to rather than included in the

form. That is, the form may refer to a previous financial report or an annual shareholders report. Therefore, in addition to a minimum document word count, we set a 200 word minimum for Item 1A, Item 7 and Item 7A.

Lastly, we require that every financial report has the signal variable available. That is, for every 10-K or 10-Q form, the corresponding company must be publicly listed on either the NASDAQ or the NYSE, with an available stock price immediately before the filing date of the financial report and exactly one quarter after the filing date. Both dates are necessary in order to calculate the change in price, as well as the change in volatility, following the release of the 10-K or 10-Q form.

The final, filtered dataset of financial reports consists of 276,880 10-K and 10-Q forms from 13,696 different companies between 2000 and 2018. In Table 1, we display some descriptives of the filtered text dataset.

	Train Dataset		**Test Dataset**	
	Positive	Negative	Positive	Negative
No. of 10-Ks	37,412	34,479	9,484	9,945
No. of 10-Qs	71,459	74,821	20,675	19,780
Avg. Word Count	44,233	41,082	50,864	54,559

Table 1: Descriptive statistics of the text dataset.

The train and test dataset is divided into a $70:30$ split, with 10% of the training dataset (approximately one year) is used for validation of model hyperparameters. The training dataset is made up of the first 14 years (2000-2013) of 10-K and 10-Q forms, amounting to 218,171 documents, of which 71,891 are 10-Ks and 146,280 10-Qs. The test dataset consists of the remaining 5 years (2014-2018), totalling to 69,884 documents, of which 19,429 are 10-Ks and 50,455 are 10-Qs. The split of positively and negatively labelled documents, in both the train and test dataset, is relatively even. Similarly, there is no clear difference in the word count of positive and negative documents, but there is an increase between the train and test dataset.

5.2 Numerical Data

Together with the database of 10-K and 10-Q forms, we extract from the CRSP and Compustat databases the stock price before and after the filing date of the respective form. The stock return is then calculated by calculating the percentage increase or decrease in stock price over the quarter of the form filing date. The stock return is used as label for document sentiment for the

10-K and 10-Q forms. Finally, the return volatility is estimated using Equation (1) .

We provide the descriptive statistics of the full dataset in Table 2. The dataset of stock returns has a mean of 0.07 with a comparatively high standard deviation at 0.79, with the quarterly return volatility estimates having a mean of 0.11 and a similarly high standard deviation of 0.82. The low mean and high standard deviation of the two variables imply that our dataset of stock returns and return volatility have a high coefficient of variation. The quantiles show that the stock returns are equally subdivided into positive and negative returns since the 50% stock return quantile is 0.00. The average word count for our dataset of 10-K and 10-Q forms is 30,600 words. Only 5% of documents have a document length smaller than 5,000 words. This supports that the financial forms are complex and lengthy reading material, and that an automatic method for measuring 10-K and 10-Q form sentiment is required.

	General Descriptives				**Quantiles**				
	Mean	Std. Dev	Max.	Min.	0.05	0.25	0.5	0.75	0.95
Stock Return	0.07	0.79	3.01	-1.00	-0.65	-0.19	0.00	0.13	0.89
Stock Return Volatility	0.11	0.82	5.46	0.00	0.00	0.01	0.03	0.08	0.28
Document Length	30600	37293	424036	2220	5673	11735	20286	38312	85269

Table 2: Descriptive statistics of the quarterly numerical data.

5.3 Parsing Text Dataset

The 10-K and 10-Q forms, retrieved from the SEC website through the EDGAR system, are downloaded as plain text files. The plain text files, in addition to containing the full text of the financial report, are outlined by a header and a tail. The header and tail of the forms include company information (found between <SEC-HEADER> and </SEC-HEADER>), metadata about the file (found between <FileStats> and </FileStats>) and tables and figures in the tail of the text. First, we scrape the needed company specific information from the header, such as the Central Index Key (CIK), 'Filed as of Date', 'Conformed Period of Report' and file type (10-K or 10-Q). The sections delimited by these tags, the table of contents and numerical tables in the appendix or text are removed as they are not meaningful in measuring sentiment.

The remaining text is the 10-K or 10-Q report. We extract each word from the plain text file and clean it by removing punctuation and capitalization. As shown in Figure 4, the form is structured in 15 separate items. However, the exact way each company separates it's 15 items is not necessarily consistent. Therefore, we construct a search algorithm that uses regular

expression (regex) to find matches to the items, such that punctuation, spaces, or non-alphabetic characters do not influence the match. The remaining text is then partitioned into separate items. We explain, in the following section how we perform our analysis on these itemized text.

6 Estimation

This section illustrates the steps we take to estimate our sentiment lexicons and the SVM and SVR procedures to forecast volatilities. First, we explain how we decide upon the set of words that are optimized through our two neural networks: the classification neural network and the regression neural network. The estimation procedure of the two feed-forward neural networks is then given, followed by the SVM and SVR model optimization and forecasting.

6.1 Defining the Lexicon

We begin by considering any word that occurs in at least one 10-K or 10-Q form; that is, all words in our dataset of financial reports. We first filter out words that arise in a negligible number of documents in our dataset. For words that appear sporadically in reports, regardless whether they influence the tone or sentiment of the text, the sentiment weight is likely to not converge to an optimal value due to the lack of observations. As done by Loughran and McDonald (2011), we take 5% as the cutoff value, which in our case amounts to 13,844 words.

Of the remaining set of words (lexicon), not all will significantly influence the sentiment of the 10-K or 10-Q form. In any language, there is a small set of commonly used words that take up a considerable percentage of the total word count. These frequently used words rarely carry any tonal weight; this list of words is referred to as stopwords. In theory, because of the high frequency of stopwords in both positive and negative documents equally, our model would be able to assign a neutral weight to stopwords. However, to facilitate the optimization of the neural networks and allow the focus to land on the important tonal words, we remove stopwords from our lexicon.

Generally, a predefined word list is used to filter out stopwords. Loughran and McDonald construct several context specific stopword lists: a general stopword list, one specific to currencies, another focused on geographical words, etc. Very little research has been carried out on the automatic filtering of stopwords. To this end, we adapt a common statistical word weighting measure, term frequency-inverse document frequency (tf-idf), to filter out stopwords.

Tf-idf is a statistical measure, defined as

$$tf\text{-}idf(t,d,D) = tf(t,d) \cdot idf(t,D),$$

where $tf(t,d)$ is a function of term frequency for term t in document d and $idf(t,D)$ is a function for inverse document frequency with D being the entire corpus of documents. In most cases, term frequency is simply fixed as $tf(t,d) = f_{t,d}$, or the frequency of term t in document d. Inverse document frequency is commonly taken as the log-scaled inverse fraction $idf(t,D) = \frac{N}{n_t}$, where N is the total number of documents $|D|$ and n_t is the number of documents term t appears in (Manning et al., 2008). Therefore, tf-idf statistic is high when n_t is low and $f_{t,d}$ is high and it is lowest when n_t is close to N.

Since stopwords appear in virtually all documents, the inverse document frequency statistic can be utilized to identify the words that are most frequently used. Therefore, we calculate the idf statistic for all words in our lexicon and remove the lowest 5% quantile.

6.2 Neural Network

Our dataset of 10-K or 10-Q forms consists of 276,880 documents, amounting to over 40GB of plain text files. Instead of using packages such as TensorFlow or Theano[5], we code our own neural network to accommodate for our extensive and unique dataset. Our implementation can be found on GitHub[6].

We code both our classification and regression feed-forward neural networks in Python without the use of deep learning packages. The general structure of the two neural networks is illustrated in Algorithm 1 below. The structure is identical for the two neural networks, with the main difference being in the forward propagation and back propagation algorithms. First, the sentiment lexicon values are initialized simultaneously with the coefficients and the hyperparameters. The sentiment positive and negative values are given random values between $[-0.25, 0.25]$. All the coefficients are initialized as 1, with the exception of the D coefficient, from Equation 5, (appearing in both neural networks) is initialized as $D = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. This is done to facilitate the optimization process of the neural network: the hyperbolic tangent layer follows the linear layer (Equation 5), and any value greater than 1 (or less than -1) inputted in

[5]TensorFlow and Theano are two Python software libraries developed for deep learning, optimization and mathematical operations, specifically for matrices.

[6]https://github.com/tgarutti/Fin-SentiLex

the *tanh* function is essentially going to output 1 (or -1), therefore hindering the backpropagation process. We train our neural network models over 5 epochs employing a batch size of 40 documents per batch.

Algorithm 1: Neural Network

Initiate: $dict \leftarrow$ initiate sentiment lexicon values between $[-0.25, 0.25]$
$\theta \leftarrow$ set initial values for coefficients
n_epochs, batch_size $\leftarrow$ set hyperparameters
Input: The training dataset of financial documents $D = \{d_1, \ldots, d_k\}$
The true signal values of each document $Y = \{y_1, \ldots, y_k\}$
Output: The optimized sentiment lexicon *dict*

for *epoch* **do**
  Randomize the order of the training dataset $\{D, Y\}$ and increment
  for $batch = \{d, y\}$ **in** dataset $\{D, Y\}$ **do**
    Define *batch* specific variables:
      $batch\_dict \leftarrow$ subset of $dict$, with only words that appear in $batch$
      $batch\_mat \leftarrow$ (weighted) position matrix

    `// Perform forward and back propagation`
    $\hat{y}, loss \leftarrow$ **ForwardPropagation**($batch$, $batch\_dict$, $batch\_mat$, θ)
    $grad\_dict, grad\_\theta \leftarrow$ **BackPropagation**($\hat{y}$, $batch$, $batch\_dict$, $batch\_mat$, θ)
    $step\_dict, step\_\theta \leftarrow$ calculate step using ADAM optimization algorithm

    `// Update parameters`
    $new\_batch\_dict \leftarrow batch\_dict - step\_dict$
    $new\_\theta \leftarrow \theta - step\_\theta$
    Use *new_batch_dict* to update sentiment lexicon *dict*

Every epoch, the dataset order is randomized in order to prevent the repetition of identical batches. For every batch, the specific batch dictionary is extracted and a weighted position matrix for each word, with respect to each document in the batch, is constructed. That is, instead of simply constructing a position matrix where for each word the frequency of occurrence per document is recorded, we utilize the more complex weight based on word frequency, average word count and inverse document frequency as defined in Equation (3).

The forward propagation algorithms for the classification and regression neural networks are detailed in Algorithm 2 and 3 respectively. The two forward propagation algorithms are analogous, following the feed-forward structures described in Sections 3.2.1 and 3.2.2. The estimate $\hat{y}$, output of the forward propagation algorithms, is utilized for the calculation of the loss functions and to obtain the gradients of the batch dictionary and of the coefficients in the back propagation algorithms.

Algorithm 2: ForwardPropagation (Classification Neural Network)

Input: Batch of dataset $batch = \{d, y\}$
Subset of sentiment lexicon, $batch\_dict$
Weighted position matrix, $batch\_mat$
Coefficients, θ

Output: $\hat{y} \leftarrow$ estimate of signal y
$loss \leftarrow$ cross entropy loss value

for doc **in** $batch$ **do**
  Retrieve weights ω of every word in doc
  Estimate $\hat{y}$ by following the layers of the neural network:
    Summation of sentiment weights (using position matrix $batch\_mat$)
    $h = \sum_{s=1}^{S} \beta_s \sum_i \omega_i$
    $\vdots$
    $\hat{y} = \left[f_\theta(\omega), \quad 1 - f_\theta(\omega) \right]^\intercal$ `// Eq.11`
$loss = -\sum_{k=1}^{K} y_k log(\hat{y}_k)$ for all K documents in $batch$

Algorithm 3: ForwardPropagation (Regression Neural Network)

Input: Batch of dataset $batch = \{d, y\}$
Subset of sentiment lexicon, $batch\_dict$
Weighted position matrix, $batch\_mat$
Coefficients, θ

Output: $\hat{y} \leftarrow$ estimate of signal y
$loss \leftarrow$ cross entropy loss value

for doc **in** $batch$ **do**
  Retrieve weights ω of every word in doc
  Estimate $\hat{y}$:
    Summation of sentiment weights (using position matrix $batch\_mat$)
    $h = \sum_{s=1}^{S} \beta_s \sum_i \omega_i$
    $\vdots$
    $\hat{y} = \delta z$ `// Eq.15`
$loss = \frac{1}{K} \sum_{k=1}^{K} (y_k - \hat{y}_k)^2$ for all K documents in $batch$

Back propagation is performed by calculating the partial derivative of the neural netowrk's loss function with respect to every parameter that needs to be optimized. The two back propagation algorithms are illustrated in Algorithm 4 and 5.

The gradients obtained from the back propagation are utilized to calculate the learning rate of stochastic gradient descent optimization algorithm. We employ the Adaptive Moment Estimation (ADAM) developed by Kingma and Ba (2015). The ADAM optimization algorithm estimates adaptive learning rates for each parameter optimized in the neural network. ADAM implements lags of the parameter gradients: both the decaying mean of lagged gradients and

Algorithm 4: BackPropagation (Classification Neural Network)

Input: Estimate of signal y, $\hat{y}$
Batch of dataset $batch = \{d, y\}$
Subset of sentiment lexicon, $batch\_dict$
Weighted position matrix, $batch\_mat$
Coefficients, θ

Output: $grad\_dict \leftarrow$ vector of gradients for each individual word in the $batch\_dict$
$grad\_\theta \leftarrow$ vector of gradients for each coefficient in neural network model

1 Calculate gradient of loss function with respect to each word ω_i in $batch\_dict$

$$\frac{\partial loss}{\partial \omega_i} = -\sum_{k=1}^{K} y_k \frac{\partial log(\hat{y}_k)}{\partial \omega_i} = -\sum_{k=1}^{K} y_k \frac{\partial log(\hat{y}_k)}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \omega_i}$$

2 Calculate gradient of loss function with respect to each coefficient j in θ

$$\frac{\partial loss}{\partial \theta^{(j)}} = -\sum_{k=1}^{K} y_k \frac{\partial log(\hat{y}_k)}{\partial \theta^{(j)}} = -\sum_{k=1}^{K} y_k \frac{\partial log(\hat{y}_k)}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \theta^{(j)}}$$

Algorithm 5: BackPropagation (Regression Neural Network)

Input: Estimate of signal y, $\hat{y}$
Batch of dataset $batch = \{d, y\}$
Subset of sentiment lexicon, $batch\_dict$
Weighted position matrix, $batch\_mat$
Coefficients, θ

Output: $grad\_dict \leftarrow$ vector of gradients for each individual word in the $batch\_dict$
$grad\_\theta \leftarrow$ vector of gradients for each coefficient in neural network model

1 Calculate gradient of loss function with respect to each word ω_i in $batch\_dict$

$$\begin{aligned}\frac{\partial loss}{\partial \omega_i} &= \frac{1}{K}\sum_{k=1}^{K} \frac{\partial (y_k - \hat{y}_k)^2}{\partial \omega_i} = \frac{1}{K}\sum_{k=1}^{K} \left(y_k^2 - 2y_k \frac{\partial \hat{y}_k}{\partial \omega_i} + \frac{\partial \hat{y}_k^2}{\partial \omega_i} \right) \\ &= \frac{1}{K}\sum_{k=1}^{K} \left(y_k^2 - 2y_k \frac{\partial \hat{y}_k}{\partial \omega_i} + 2\hat{y}_k \frac{\partial \hat{y}_k}{\partial \omega_i} \right)\end{aligned}$$

2 Calculate gradient of loss function with respect to each coefficient $\theta^{(j)}$ in θ

$$\begin{aligned}\frac{\partial loss}{\partial \theta^{(j)}} &= \frac{1}{K}\sum_{k=1}^{K} \frac{\partial (y_k - \hat{y}_k)^2}{\partial \theta^{(j)}} = \frac{1}{K}\sum_{k=1}^{K} \left(y_k^2 - 2y_k \frac{\partial \hat{y}_k}{\partial \theta^{(j)}} + \frac{\partial \hat{y}_k^2}{\partial \theta^{(j)}} \right) \\ &= \frac{1}{K}\sum_{k=1}^{K} \left(y_k^2 - 2y_k \frac{\partial \hat{y}_k}{\partial \theta^{(j)}} + 2\hat{y}_k \frac{\partial \hat{y}_k}{\partial \theta^{(j)}} \right)\end{aligned}$$

the decaying average of past square gradients, which can be seen as a proxies for the mean and variance respectively. The ADAM method follows,

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned}$$

to evaluate the exponentially decaying averages. Since the m_t and v_t moment estimates are initialized as vectors of zero's, the estimates are corrected for the zero bias:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2}. \end{aligned}$$

The learning rate is then calculated and the parameters updated by means of the ADAM update procedure:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t.$$

As suggested by Kingman and Ba, we set the parameters of the ADAM optimization algorithm as $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 1 \times 10^{-8}$.

We train our neural network based sentiment lexicons over 5 epochs following this estimation procedure. All resulting words in the two lexicons have a positive and negative value; the score of each word can be evaluated as the difference between the two values. We filter our lexicons further by removing the neutral terms, or the terms with a score closest to zero. We do this by removing, from each lexicon, the 20% percentile of words with absolute score closest to zero.

6.3 Support Vector Machines

We estimate our Support Vector Machines (SVMs) and Support Vector Regression Machines (SVRs) utilizing the scikit-learn machine learning library for python. In this paper, we evaluate the comparative performance of our automatically constructed sentiment lexicons and the benchmarks by integrating the lexicons in SVM and SVR based models. The classification SVM models are employed to forecast two categorical variables: an indicator for change in stock return (1 if positive and 0 if negative) and the trend volatility variable v_k (1 if positive and 0 if negative), explained in Section 3.3.1. The SVR models are utilized to forecast the actual stock return volatility value.

We use 10% of our training dataset as a validation dataset to optimize the hyperparameters of the SVM and SVR models for each variable by performing a grid search over several hyperparamters. In the grid search, we focus on the kernel, as it is the function used to transform the feature space and is critical to the model's performance, and gamma, which determines how influential each training data sample is with regard to the optimization of the model. The grid search iterates over four kernels, linear, 3rd degree polynomial, RBF and sigmoid, as well as six different gamma variables, 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, $auto = \frac{1}{Size(\mathbf{x}_k)}$, and $scale = \frac{1}{Size(\mathbf{x}_k) + V(\mathbf{x}_k)}$, where $\mathbf{x}_k$ is the input features matrix of the SVM or SVR model, $Size(\mathbf{x}_k)$ is the number of features and $V(\mathbf{x}_k)$ is the mean of the variances of each feature in $\mathbf{x}_k$. We employ accuracy - i.e, the percentage of correct classifications - to evaluate the SVM models,

and root mean square forecast error for the SVR models.

The results of the grid searches are displayed in the figures below. Figure 5 shows the accuracy results of the grid search performed on the change in stock return indicator variable with SVMs integrated with our classification and regression lexicons respectively. It is evident that the sigmoid kernel outperforms the rest of the kernels, specifically for *auto* and *scale* gamma values.

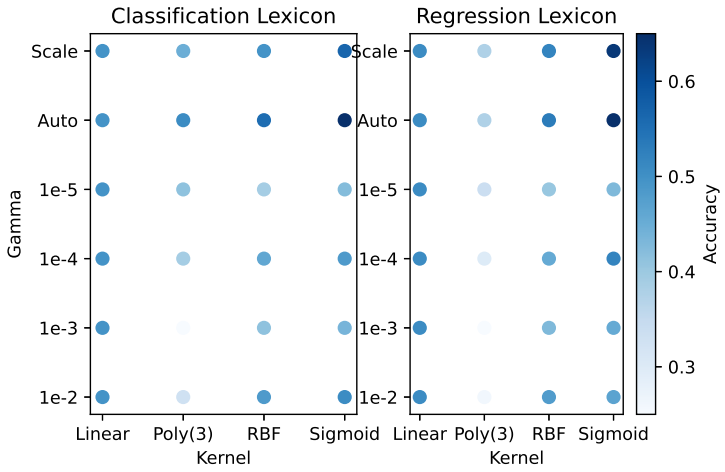

Figure 5: Grid search accuracy results for hyperparameters kernel and gamma in lexicon integrated SVMs, considering the change in stock return indicator variable. Accurcay is displayed as the shade of each dot (darker implying higher accuracy).

In parallel with the above results, the results for the grid search performed on the stock return volatility change trend variable show higher accuracy for the sigmoid kernel and for *auto* and *scale* gamma values. The RBF kernel, however, seems to perform similarly, in terms of accuracy, for the classification lexicon integrated SVM.

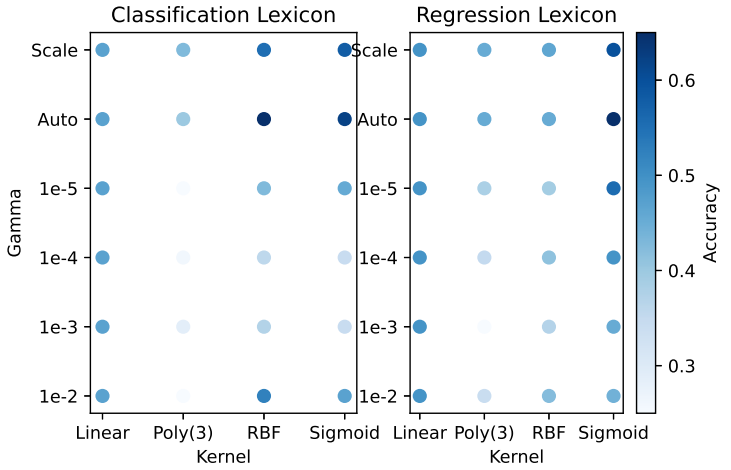

Figure 6: Grid search accuracy results for hyperparameters kernel and gamma in lexicon integrated SVMs, considering the stock return volatility trend variable. Accurcay is displayed as the shade of each dot (darker implying higher accuracy).

Since the SVR models forecast numerical values, the accuracy is measured by RMSE. The lowest RMSE is given by the RBF kernel with the *auto* and *scale* gamma values. The 3rd degree polynomial and sigmoid kernels seem to have just lower RMSE values for the *auto* gamma value.

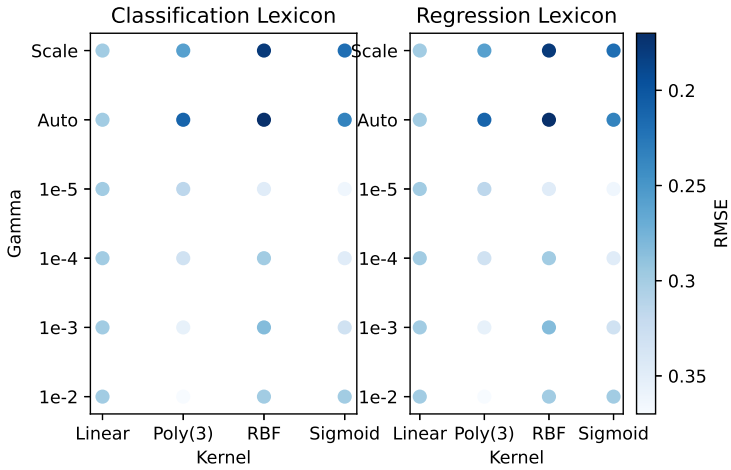

Figure 7: Grid search RMSE results for hyperparameters kernel and gamma in lexicon integrated SVRs, considering the stock return volatility. The RMSE measure is displayed as the shade of each dot (darker implying lower RMSE).

Therefore, following the results of the grid search, we estimate our SVM models employing the sigmoid kernel and *scale* gamma value, while the RBF kernel and *scale* gamma value are used for SVR models.

7 Results

In this section we present the results of our analysis. First, we display some information on the two sentiment lexicons resulting from the neural network optimization. We then forecast change in volatility and volatility, employing the SVM and SVR models in conjunction with the sentiment lexicons. Statistical measures are implemented to evaluate the performance of our models in comparison to the benchmarks.

7.1 Sentiment Lexicons

The classification and regression based feed-forward neural networks, described in Section 3.2.1 and Section 3.2.2 respectively, extract sentiment regarding the financial status of a company from their 10-K and 10-Q forms. Each sentiment carrying word is represented in our neural network by a two-dimensional vector. This two-dimensional word representation consists of a positive and a negative sentiment score, $w_i = (n_i, p_i)$. The sentiment score or weight of a word is calculated as the difference between the positive score p_i and negative score n_i. We provide some statistics on the structure of the sentiment lexicons in Table 3.

	Positive Words	Negative Words	Total
Loughran & McDonald	354	2353	2707
Vo & Zhang	6205	6281	12486
Classification Fin-SentiLex	954	998	1952
Regression Fin-SentiLex	1571	381	1952

Table 3: Statistics regarding the number of positive, negative words in the respective sentiment lexicons.

Our two neural networks, Classification and Regression Fin-SentiLex, consist of the fewest number of sentiment words, while the neural network based lexicon, developed by Vo and Zhang (explained in Section 4.2), is made up of over 12,000 sentiment words. Both the Vo and Zhang Benchmark and our Classification sentiment lexicon have an even number of positively and negatively weighted words. For the Regression lexicon, however, only 20% of all sentiment words are negatively scored, with the remaining 80% being positive. Oppositely, the Loughran dictionary has a large number of negative sentiment words.

To provide further insight into the composition of our two sentiment lexicons, we show the wordlists of the highest scored positive words and lowest scored negative words for the Classification Fin-SentiLex, in Table 4, and the Regression Fin-SentiLex in Table 5. It is apparent

from the wordlists that the two neural network formulations are able to reasonably classify sentiment. This is emphasized by positively weighted words, such as 'profitably', 'growing', 'confirmation' and 'reinvest', and negative words, such as 'penalty', 'interference', 'minimum' and 'instability'. Furthermore, there are several financial terms appearing in the wordlists: 'debenture', 'equity' and, most notably, 'taxable' and 'taxation', that appear in the positive wordlist and negative wordlist of Table 4, respectively.

Positive	No. of Documents	Negative	No. of Documents
Clarification	16,332	Handled	15,627
Taxed	20,676	Debenture	16,047
Apart	21,022	Flood	17,613
Profitably	22,945	Ascertain	18,505
Willingness	24,100	Wrongful	18,577
Composite	25,513	Unfair	25,275
Oil	45,280	Unconditionally	29,325
Vigorously	55,376	Slowdown	33,759
Reimburse	61,477	Climate	35,300
Subjective	73,091	Instability	36,502
Accurately	81,897	Minor	37,961
Expanded	87,337	Taxation	48,210
Enable	104,600	Uncollectible	53,661
Award	118,563	Appear	57,861
Taxable	121,339	Accrual	88,152
Consistent	148,236	Repay	91,576
Support	165,767	Damages	121,478
Complete	165,810	Least	158,933
Corporate	177,856	Minimum	159,751
Equity	203,040	Necessary	206,695

Table 4: Wordlist of top negative and positive sentiment words in the Classification Fin-SentiLex. Next to each word, the respective number of documents, that the word appears in at least once, is given.

An apparent difference between the positive wordlists and the negative wordlists in both Table 4 and Table 5 is the number of documents a word appears in. There seems to be a disparity between the positive and negative wordlists, with positively scored words generally occurring in a higher number of documents than negative words. In our feed-forward neural networks, we utilize a weighting scheme based on inverse document frequency. Since the negative words in the two tables generally have a lower number of documents, it is likely that they receive a higher weight in our weighting scheme. Furthermore, the wordlists of the Classification and Regression sentiment lexicons are very different, with no words in common between the two tables. In the next section, we evaluate the forecasting performance of the SVM and SVR

models based on the sentiment lexicons.

Positive	No. of Documents	Negative	No. of Documents
Reinvest	19,774	Debenture	16,047
Induce	25,658	Sue	16,500
Optional	28,534	Disclose	16,500
Confirmation	29,824	Subordination	18,711
Leadership	32,273	Verification	19,949
Enough	38,841	Denial	20,628
Suitable	59,175	Resign	22,252
Forecast	63,394	Interference	24,317
Growing	71,952	Compromise	24,761
Recover	84,980	Viability	26,364
Individually	90,121	Feasibility	31,285
Supporting	97,226	Irrevocably	31,481
Achieve	127,609	Delinquent	33,568
Comprised	132,670	Unrestricted	34,208
Favorable	136,149	Irrevocable	42,757
Adequate	162,185	Priority	61,498
Significantly	176,102	Penalty	63,025
Fixed	178,472	Removed	74,741
Economic	186,493	Notwithstanding	78,825
Consolidated	204,652	Payroll	79,725

Table 5: Wordlist of top negative and positive sentiment words in the Regression Fin-SentiLex. Next to each word, the respective number of documents, that the word appears in at least once, is given.

7.2 Forecasting Results

In this section, we assess the lexicons by means of lexicon integrated deep learning methods, and forecasting evaluation measures based on forecasts of three different variables. We forecast two categorical variables employing the GARCH-based SVM method, explained in Section 3.3.1, and forecast one numerical variable by means of the GARCH-based SVR method from Section 3.3.2.

The first categorical variable we investigate is the stock return categorical variable used as a sentiment label in the optimization and learning of the sentiment lexicons. The stock return indicator variable, r_k takes value 1 if the change in stock return of document d_k is positive - that is, if the stock price one quarter after the filing of the 10-K or 10-Q form of the respective company is greater than the stock price at the time of the filing - and takes value 0 if the change in stock return is negative. The second variable is the volatility indicator variable v_k, which similarly takes value 1 if the change in return volatility, over the filing period of the financial

report, is positive, and 0 if the change is negative. Finally, the continuous variable considered is the actual stock return volatility σ_k^2 one period after the filing of the document d_k.

In the field of machine learning classification, the metrics needed to evaluate the forecasting performance of a classification model are based on the confusion matrix. The confusion matrix is a table that displays the performance of a classification model. As displayed in the matrix below, each column of the matrix is an instance of the predicted class, the rows are instances of the true class, and each cell represents the corresponding number of observations.

		Predicted Class	
		Class = 1	Class = 0
True Class	Class = 1	TP	FN
	Class = 0	FP	TN

In the analysis of our classification models, we evaluate the performance based on the precision, recall, F1 and accuracy measures. Precision is the ratio of correctly class predictions over the total number of predictions of that class. That is, for class 1, precision is calculated as,

$$Precision = \frac{TP}{TP+FP}.$$

Recall is measured as the number of predicted class 1, divided by the total number of observations of class 1:

$$Recall = \frac{TP}{TP+FN}.$$

The F1 statistic is estimated as a weighted average of the two metrics, precision and recall,

$$F1 = 2\,\frac{Recall \times Precision}{Recall + Precision}.$$

The three measures, precision, recall and F1, are estimated for both class 1 and class 0. Finally, the accuracy statistic is the ratio of correct predictions over the total number of predictions:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}.$$

The performance measures for the forecasting results of the stock returns categorical variable are evaluated over the complete train and test dataset. In Table 6, we display the results for the

support-vector machine methods integrated with our automatically constructed sentiment lexicons, Classification Fin-SentiLex and Regression Fin-SentiLex, and the benchmark lexicons, Loughran and Vo and Zhang and a simple baseline. This baseline finds the majority class, 1 or 0, of the training data and strictly forecasts that class.

	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.5090	-	-	1	0	-	0.6697	-	-	0.5090
Loughran & McDonald	0.4957	0.6582	0.5770	0.6182	0.5390	0.5786	0.5502	0.5927	0.5714	0.5725
Vo and Zhang	0.4469	0.6036	0.5252	0.5567	0.4948	0.5258	0.4958	0.5438	0.5198	0.5210
Classification Fin-SentiLex	0.4735	0.6316	0.5526	0.5816	0.5260	0.5538	0.5220	0.5740	0.5480	0.5495
Regression Fin-SentiLex	0.4541	0.5848	0.5195	0.2163	0.8094	0.5128	0.2930	0.6790	0.4860	0.5585

Table 6: Return indicator variable r_k forecasting results (precision recall and F1 statistic). Training and forecast evaluation performed using the complete train and test sample.

From Table 6, it is evident that there is a disparity in precision for class 1 (positive stock returns) and class 0 (negative stock returns). This disparity, where precision is greater for class 0, is present in all four classification models. This implies that the lexicon integrated SVM models have fewer false predictions for negative stock returns than for positive stock returns. This result is in line with what was hypothesized by Loughran and McDonald (2011): that the language of financial reports, such as 10-K and 10-Q forms, have a primarily negative effect on a company's future stock returns.

In contrast, we find that the recall statistic has the opposite imbalance for all SVMs apart for the Regression lexicon integrated SVM. A higher recall for class 1 means that the percentage of positive returns correctly classified by the lexicon based classification method is higher than the percentage of negative returns. The Regression Fin-SentiLex method is the only SVM model that correctly classifies a significant proportion of negative stock returns.

The F1 statistic is a weighted average of precision and recall. Therefore, it follows from the disparities between the classes in the precision and recall measures, that, for the first three lexicon integrated SVMs, there is parallelism between the F1 statistic of each class.

Analysing the model forecast accuracy in addition to the averages for precision, recall and F1 statistic, it is evident that the Loughran and McDonald based SVM model outperforms the other lexicon integrated SVM models in foreacsting the sign of a company's future stock return. Out of the SVM models employing automatically constructed lexicons (Vo and Zhang, Classification Fin-SentiLex and Regression Fin-SentiLex), our Regression Fin-SentiLex based model outperforms the Vo and Zhang lexicon by almost 4% in terms of accuracy. However, considering the precision, recall and F1 measures, the Vo and Zhang based model has higher values

for every measure. This is likely due to the large difference between the measures for class 1 and class 0 and the inability of the Regression Fin-SentiLex based model to accurately forecast positive stock returns (class 1). The SVM model employing the Classification Fin-SentiLex outperforms the Vo and Zhang lexicon in accuracy, precision, recall and F1 statistic. All models perform better than the baseline model in terms of accuracy.

In addition to performing the analysis over the complete train and test datasets, we analyse the forecasting performance of our lexicon integrated models over several subsamples of the datasets. We do this in order to evaluate the learning ability and forecast performance of the lexicon based SVM models with smaller datasets and as a robustness check to ascertain whether the full-sample results displayed in Table 6 are consistent.

Subsample 1: Train 10,000, Test 2,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	-	0.5000	-	0	1	-	-	0.6667	-	0.5000
Loughran & McDonald	0.4357	0.5417	0.4887	0.6893	0.2915	0.4904	0.5339	0.3790	0.4895	0.4675
Vo & Zhang	0.5227	0.5970	0.5599	0.3898	0.7175	0.5537	0.4466	0.6517	0.5567	0.5725
Classification Fin-SentiLex	0.6154	0.6407	0.6281	0.4520	0.7758	0.6139	0.5212	0.7018	0.6209	0.6325
Regression Fin-SentiLex	0.5274	0.6063	0.5668	0.4350	0.6906	0.5628	0.4768	0.6457	0.5648	0.5775
Subsample 2: Train 20,000, Test 4,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.5010	-	-	1	0	-	0.6676	-	-	0.5010
Loughran & McDonald	0.5266	0.5990	0.5628	0.2553	0.8290	0.5421	0.3438	0.6955	0.5523	0.5848
Vo & Zhang	0.5251	0.6005	0.5628	0.2693	0.8185	0.5439	0.3560	0.6928	0.5532	0.5840
Classification Fin-SentiLex	0.5189	0.5977	0.5583	0.2576	0.8220	0.5398	0.3443	0.6921	0.5489	0.5814
Regression Fin-SentiLex	0.5190	0.5975	0.5583	0.2553	0.8237	0.5395	0.3422	0.6926	0.5487	0.5812
Subsample 3: Train 30,000, Test 6,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.4795	-	-	1	0	-	0.6482	-	-	0.4795
Loughran & McDonald	0.4795	0.6425	0.5610	0.6087	0.5156	0.5622	0.5365	0.5721	0.5616	0.5750
Vo & Zhang	0.4331	0.6087	0.5209	0.7766	0.2548	0.5157	0.5561	0.3592	0.5183	0.4755
Classification Fin-SentiLex	0.4833	0.6615	0.5724	0.6667	0.4775	0.5721	0.5604	0.5546	0.5722	0.5575
Regression Fin-SentiLex	0.4681	0.6374	0.5528	0.6336	0.4723	0.5529	0.5384	0.5426	0.5528	0.5405
Subsample 4: Train 50,000, Test 10,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	-	0.49225	-	0	1	-	-	0.6597	-	0.4923
Loughran & McDonald	0.5128	0.6763	0.5945	0.7836	0.3777	0.5807	0.6199	0.4847	0.5875	0.5958
Vo & Zhang	0.4801	0.5838	0.5320	0.6442	0.4172	0.5307	0.5502	0.4866	0.5313	0.5205
Classification Fin-SentiLex	0.5104	0.6592	0.5848	0.7567	0.3933	0.5750	0.6096	0.4927	0.5799	0.5588
Regression Fin-SentiLex	0.5817	0.6021	0.5919	0.3987	0.7604	0.5796	0.4731	0.6721	0.5857	0.5625

Table 7: Return indicator variable r_k forecasting results (precision recall and F1 statistic). Training and forecast evaluation performed using four different subsamples of the train and test datasets.

The full training dataset consists of 218,171 financial documents and the full test datasets is made up of 59,884 documents. To carry out the robustness, we randomly select a subsample of the training dataset, e.g., 10,000 financial documents, and evaluate the forecasting performance over a subsample of the test dataset, e.g., 2,000 documents. For a specific subsample dimension

(train 10,000 and test 2,000), we perform the forecast evaluation 20 times, each time randomly selecting the subsample of train and test datasets from the complete dataset, and calculate the mean of the precision, recall, F1 and accuracy measures. We conduct this procedure for four different subsample dimensions: train 10,000 and test 2,000; train 20,000 and test 4,000; train 30,000 and test 6,000; train 50,000 and test 10,000. The averages of the forcast performance measures for each subsample are displayed in Table 7.

Apart from the first subsample, the results from the robustness check seem to be generally in line with those of the full sample forecast evaluation from Table 6. The same inequality in class 1 and class 2 precision remains present in every subsample. Our Classification and Regression Fin-SentiLex integrated SVMs outperform, on average, the Vo and Zhang benchmark SVM model for subsamples 1, 3 and 4. However, the difference in forecasting performance of the four models in subsample 2 is negligible, with all four models having almost identical accuracies, precision, recall and F1 statistic. The Loughran and McDonald based SVM generally performs best in terms of accuracy. However, in subsample 1, the accuracy measure of the Loughran and McDonald based SVM is the lowest out of all models, including the majority baseline. This could be explained by the fact that the Loughran and McDonald sentiment lexicon was not automatically constructed over our dataset of 10-K and 10-Q forms, but manually built considering a different dataset of 10-Ks and 10-Qs. Therefore, a small subsample of 10,000 financial forms could have a lower frequency of sentiment carrying words from the Loughran and McDonald lexicon, leading to inaccurate forecasts. For subsample 1, the Classification lexicon based SVM clearly outperforms the rest, with regard to all forecast evaluation measures.

	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	-	0.5050	-	0	1	-	-	0.6711	-	0.5050
SVM Baseline	0.5892	0.4234	0.5063	0.4870	0.5259	1.0129	0.5332	0.4691	0.5012	0.5063
Loughran & McDonald	0.5667	0.5753	0.5710	0.5736	0.5685	0.5710	0.5701	0.5719	0.5710	0.5710
Vo & Zhang	0.5027	0.5244	0.5135	0.7641	0.2560	0.5100	0.6064	0.3440	0.4752	0.5080
Classification Fin-SentiLex	0.5331	0.5776	0.5553	0.7147	0.3839	0.5493	0.6107	0.4613	0.5360	0.5480
Regression Fin-SentiLex	0.5500	0.6064	0.5782	0.7258	0.4157	0.5707	0.6258	0.4932	0.5595	0.5695

Table 8: Volatility indicator variable v_k forecasting results (precision recall and F1 statistic). Training and forecast evaluation performed using the complete train and test sample.

Next, we examine the sentiment lexicon integrated SVM models' forecasts of the categorical variable for the change in stock return volatility: class 1 represents an increase in stock return volatility, while class 0 represents a decrease. We present the forecast results for the full train and test dataset in Table 8 below. In addition to the simple majority baseline, we include the SVM baseline that forecasts without the integration of a lexicon. That is, the SVM baseline

takes only the past volatility and shock terms as input features without the document sentiment feature.

Firstly, we note, from Table 8, that the precision statistics comparable across the two classes for all lexicon integrated SVMs. However, considering the recall statistic, there is a clear tendency towards class 1 for all SVMs based on automatically constructed lexicons. This implies that these SVMs more accurately forecast increases in stock return volatility than decreases. Since we remove words with a 'neutral' score (score close to zero) from our lexicons, this result is expected. In removing neutral words, the lexicons are left with either highly negative or highly positive words. Words with high sentiment scores appear more frequently in documents that similarly carry significant sentiment, either negative or positive, that is likely to induce an increase in stock return volatility.

Observing the accuracy statistic, the Loughran and McDonald integrated SVM has the highest accuracy at 0.5710, only just higher than the Regression SVM with 0.5695. These two SVMs outperform the rest of the SVMs in precision, recall and F1. All lexicon based SVMs outperform the SVM baseline and the majority baseline, implying that the integration of the sentiment lexicon in the SVM models improves its accuracy. We further investigate the forecasting performance of the SVM models by conducting the same subsample robustness check.

The results of the robustness check are displayed in Table 9. A notable difference in the forecast evaluation measures in Table 9, is that for the first two subsamples the recall statistic has the opposite tendency than for the full-sample measure in Table 8. This difference in results could be explained by the small sample sizes. With a small sample, there is a higher chance of having an uneven class distribution; that is, if there is a greater number of financial documents preceding a decrease in stock return volatility (class 0) than an increase in the volatility (class 1), the SVM could be biased towards forecasting the more frequent class. This is supported by the fact that, for subsample 3 and subsample 4, the tendency of the recall statistic reverts back to class 1.

Subsample 1: Train 10,000, Test 2,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	-	0.4962	-	0	1	-	-	0.6632	-	0.4962
SVM Baseline	0.5372	0.5090	0.5231	0.3218	0.7172	1.0390	0.4025	0.5954	0.4989	0.5175
Loughran & McDonald	0.7064	0.5724	0.6394	0.3831	0.8384	0.6107	0.4968	0.6803	0.6247	0.6090
Vo & Zhang	0.5584	0.5077	0.5331	0.2129	0.8283	0.5206	0.3082	0.6296	0.5268	0.5175
Classification Fin-SentiLex	0.5889	0.5195	0.5542	0.2637	0.8122	0.5379	0.3643	0.6337	0.5459	0.5352
Regression Fin-SentiLex	0.6923	0.5608	0.6266	0.3564	0.8384	0.5974	0.4706	0.6721	0.6116	0.5950
Subsample 2: Train 20,000, Test 4,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.4950	-	-	1	0	-	0.6622	-	-	0.4950
SVM Baseline	0.5093	0.5232	0.5163	0.3906	0.6399	1.0305	0.4421	0.5757	0.5089	0.5180
Loughran & McDonald	0.6435	0.5827	0.6131	0.4172	0.7789	0.5980	0.5062	0.6667	0.6055	0.6020
Vo & Zhang	0.5764	0.5370	0.5567	0.2699	0.8102	0.5401	0.3677	0.6459	0.5482	0.5460
Classification Fin-SentiLex	0.5990	0.5380	0.5685	0.2413	0.8454	0.5434	0.3440	0.6575	0.5556	0.5500
Regression Fin-SentiLex	0.6471	0.5809	0.6140	0.4057	0.7882	0.5970	0.4987	0.6689	0.6054	0.6012
Subsample 3: Train 30,000, Test 6,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.4998	-	-	1	0	-	0.6665	-	-	0.4998
SVM Baseline	0.4794	0.4917	0.4856	0.4103	0.5615	0.9718	0.4422	0.5243	0.4832	0.4865
Loughran & McDonald	0.5991	0.6821	0.6406	0.7651	0.4960	0.6306	0.6720	0.5744	0.6355	0.6295
Vo & Zhang	0.5144	0.5849	0.5497	0.8448	0.2153	0.5300	0.6395	0.3147	0.5397	0.5275
Classification Fin-SentiLex	0.5392	0.6351	0.5872	0.8175	0.3125	0.5650	0.6498	0.4189	0.5759	0.5630
Regression Fin-SentiLex	0.5709	0.5771	0.5740	0.5685	0.5794	0.5740	0.5697	0.5782	0.5740	0.5740
Subsample 4: Train 50,000, Test 10,000										
	Precision			Recall			F1			
	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	Class = 1	Class = 0	Macro-Avg.	**Accuracy**
Baseline	0.5012	-	-	1	0	-	0.6677	-	-	0.5012
SVM Baseline	0.5613	0.4784	0.5198	0.4752	0.5644	1.0396	0.5147	0.5178	0.5162	0.5163
Loughran & McDonald	0.6371	0.6350	0.6360	0.7581	0.4935	0.6258	0.6923	0.5554	0.6309	0.6363
Vo & Zhang	0.5616	0.5214	0.5415	0.7669	0.2978	0.5324	0.6484	0.3791	0.5369	0.5510
Classification Fin-SentiLex	0.5927	0.5740	0.5834	0.7493	0.3962	0.5728	0.6619	0.4688	0.5780	0.5868
Regression Fin-SentiLex	0.6374	0.5470	0.5922	0.5553	0.6295	0.5924	0.5936	0.5854	0.5923	0.5895

Table 9: Volatility indicator variable v_k forecasting results (precision recall and F1 statistic). Training and forecast evaluation performed using four different subsamples of the train and test datasets.

The forecasting performance, in terms of accuracy of the model, is relatively consistent throughout the subsamples. As in the results for the complete train and test dataset, the Loughran and McDonald lexicon based SVM is the model that forecasts the stock return volatility categorical variable with highest accuracy. However, the performance of the SVM based on the Regression Fin-SentiLex is comparable, with a difference of 1% or less in accuracy for subsample 1 and 2, as well as for the full dataset. In all of the subsamples the two baselines, the majority and SVM baselines, have the lowest accuracy. This once again shows the implementation of the sentiment lexicon increases the accuracy of the forecast of the volatility indicator variable.

Finally, we forecast the numerical variable, stock return volatility, by means of lexicon integrated Support Vector Regression models (SVRs). Since stock retrun volatility is a continuous variable, we evaluate the forecasts by means of root mean square error (RMSE). We present

the RMSE values for the full train and test dataset forecasts, as well as the robustness check subsample results, in Table 10. As done before, we include the SVR baseline that takes only the past volatility and shock terms as input features and doesn't include the document sentiment feature.

	RMSE				
	Subsample 1	Subsample 2	Subsample 3	Subsample 4	Full Sample
SVR Baseline	0.2619	0.2114	0.2047	0.1994	0.2201
Loughran & McDonald	0.2501	0.1952	0.1926	0.1910	0.2149
Vo & Zhang	0.2594	0.2059	0.2019	0.1992	0.2121
Classification Fin-SentiLex	0.2516	0.1966	0.1930	0.1927	0.2090
Regression Fin-SentiLex	0.2510	0.1944	0.1902	0.1878	0.2081

Table 10: Volatility σ_k^2 forecasting results (RMSE). Training and forecast evaluation performed using the complete train and test sample as well as four different subsamples.

The mean RMSEs for subsample 1 show that a subsample of only 10,000 training observations does not have enough labels to correctly learn and forecast the stock return volatility. The significantly lower RMSEs for the other subsamples and the full sample further supports this. Furthermore, the fact that the mean RMSEs of subsamples 2, 3 and 4 are lower than the RMSEs of the full sample implies that learning over such an extensive training dataset might be detrimental to the forecasting performance. All lexicon integrated SVR models outperform the SVR baseline, which suggests that the inclusion of a sentiment lexicon based feature enhances the forecast of the stock return volatility. The Regression Fin-SentiLex integrated SVR has the comparatively best stock return volatility forecasting performance, with the lowest average RMSE in subsample 2, 3 and 4. Additionally, the Regression Fin-SentiLex based SVR has the best forecasting performance for the complete train and test dataset. This is likely due to the fact that the model was trained based on a regression neural network, therefore having continuous variables as labels to train the lexicon instead of categorical variables. This allows for a more accurate sentiment lexicon learning.

However, we note that the RMSE values are high, considering that stock return volatility has a mean of 0.11 with a standard deviation of 0.82 (Table 2). Furthermore, the accuracy of our SVM models, on average between 0.55 and 0.60, are quite low. We hypothesize that this is due to the structure of the dataset used to perform our analysis. Specifically, we theorize that the length of the 10-K and 10-Q forms utilized to learn the sentiment lexicons has severely influence the optimization performance of the model. In general, automatic sentiment lexicon construction is conducted on short texts (Vo and Zhang, 2016; Li and Shah, 2017; Tang et al.,

2014b), such as tweets or titles of news articles. This is done to optimize the learning of the lexicon by having a small number of significant words classified by a sentiment label. In our case, each 10-K or 10-Q has one label: in the case of the classification neural network based lexicon the label is $[1,0]$ when the change in respective stock price is positive and $[0,1]$ if negative; in the case of the regression neural network based lexicon, the label is the respective stock return. Therefore, we employ one label for a financial text with, on average, 30,000 words. This lack of sentiment signal in the optimization and training process of the neural networks significantly complicates the learning of the sentiment lexicons, and can, to a certain extent, explain the low accuracy values and the slight outperformance of the Loughran and McDonald manually constructed sentiment lexicon.

Nevertheless, by comparatively analysing the results it is evident that our two automatically constructed sentiment lexicons, the classification and regression neural network based lexicons, are not able to outperform the Loughran and McDonald manually constructed lexicon for the classification of the two categorical variables by means of machine learning techniques. The forecasting accuracy, of the SVMs based on our lexicons, is higher than that of the SVM based on the Vo and Zhang automatically constructed lexicon for both categorical variables. In additon, the forecasting of the continuous variable of the Regression Fin-SentiLex integrated SVR provides smaller RMSEs than any other lexicon integrated SVR.

8 Conclusion

In this paper, we investigate the sentiment of an extensive financial corpora of 10-K and 10-Q reports. We develop two methods to automatically construct a finance-specific sentiment lexicon based on the financial reports. The first is a classification feed-forward neural network framework utilizing an indicator for change in a company's stock returns as sentiment label for the corresponding company's 10-K or 10-Q. While the second is regression feed-forward neural network framework that employs the actual change in a company's stock return as sentiment label. In addition to constructing the sentiment lexicons, machine learning techniques are utilized to evaluate the performance of the lexicons. We construct GARCH-based SVM and SVR models, integrated with our sentiment lexicons, to forecast stock return volatility and classify trend variables for change in stock returns and change in stock return volatility.

The results of our forecast evaluation show that our automatically constructed sentiment

lexicons are not able to outperform Loughran and McDonald's manually constructed sentiment lexicon in either classifying the change in stock returns or the change in stock return volatility. The robustness check shows that the Loughran and McDonald lexicon integrated SVM models have a higher accuracy in classification for almost all subsamples. Nevertheless, our two lexicons perform better than Vo and Zhang's automatically constructed lexicon and the baseline. In addition, the Regression Fin-SentiLex integrated SVR model forecasts stock return volatility more accurately than all lexicons, including Loughran and McDonald's manually constructed lexicon.

However, the margin of outperformance in RMSEs and classification accuracy is small, and the performance of all SVM and SVR models is unsatisfactory with accuracy values generally at 60% or lower and relatively high RMSE values. This underlines a significant limitation in our study. This low accuracy is likely to be due to the structure of the financial corpora that we base our research on. Our dataset is comprised of 10-K and 10-Q forms, which, on average, consisting of 30,000 words. Since each document has exactly one sentiment label, based on stock return, the sentiment signal each word is subject to, in the training of the lexicon, can be considered imprecise. This could significantly hinder the optimization process and therefore lead to inaccurate sentiment lexicons. Furthermore, the general inaccuracy of the SVM and SVR models could be further explained by the size of the dataset, since the performance on the smaller subsamples is comparable to the full sample, or occasionally better.

Further research on sentiment analysis of 10-K and 10-Q forms is therefore needed. An interesting avenue of research could entail employing the regression neural network formulation to learn a sentiment lexicon based on shorter texts, such as stocktweets or news headlines, and apply the learned lexicon on 10-K and 10-Q reports. However, this might prove challenging as the language used in tweets and titles might not be transferable to regulated financial reports. Additionally, the filtering of the 10-K and 10-Q forms could be improved upon. Since the forms consist of 30,000 words, a method for a comprehensive filtering of financial forms could be developed, with the aim of increasing the impact of the sentiment labels on specific words. This could be done by focusing on specific Items of the 10-K and 10-Q forms that are expected to significant textual information (Item 1A, Item 7 and Item 7A). Alternatively, a method for filtering out the neutral words from the learned sentiment lexicon could enhance the accuracy of the lexicon and therefore any lexicon integrated models. One could develope a machine learning technique to filter a sentiment lexicon after the learning process. That is, more accurately

removing neutral words from the lexicon and therefore adding importance to the remaining sentiment carrying words. Hence, while this research has its limitations, the automatic construction of sentiment lexicons based on 10-K and 10-Q reports should be investigated further.

References

Amir, S., Astudillo, R. F., Ling, W., Carvalho, P. C., and Silva, M. J. (2017). Expanding subjective lexicons for social media mining with embedding subspaces. *arXiv Preprint*, arXiv:1701.00145.

Astudillo, R., Amir, S., Ling, W., Silva, M., and Trancoso, I. (2015). Learning word representations from scarce and noisy data with embedding subspaces. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084. ACL.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. ELRA.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327.

Bonsall, S. B., Leone, A. J., Miller, B. P., and Rennekamp, K. (2017). A plain english measure of financial reporting readability. *Journal of Accounting and Economics*, 63(2):329–357.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT 1992, page 144–152. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Davidov, D., Tsur, O., and Rappoport, A. (2010). Enhanced sentiment learning using Twitter hashtags and smileys. pages 241–249. 23rd International Conference on Computational Linguistics (COLING 2010).

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., and Vapnik, V. (1996). Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 155–161, Cambridge, MA, USA. MIT Press.

Esuli, A. and Sebastiani, F. (2005). Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, (CIKM 2005), page 617–624. ACL.

Esuli, A. and Sebastiani, F. (2006a). Determining term subjectivity and term orientation for opinion mining. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 193–200. ACL.

Esuli, A. and Sebastiani, F. (2006b). SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*. ELRA.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *35th Annual Meeting of the Association for Computational Linguistics*, pages 174–181. ACL.

Kamps, J., Marx, M., Mokken, R. J., and de Rijke, M. (2004). Using WordNet to measure semantic orientations of adjectives. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. ELRA.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (CICLR 2015)*.

Kouloumpis, E., Wilson, T., and Moore, J. (2011). Twitter sentiment analysis: The good the bad and the OMG! In *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICWSM*, pages 538–541. AAAI Press.

Li, Q. and Shah, S. (2017). Learning stock market sentiment lexicon and sentiment-oriented word vector from stocktwits. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 301–310. ACL.

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

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013)*, page 3111–3119. Curran Associates Inc.

Mohammad, S., Kiritchenko, S., and Zhu, X. (2013). NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327. ACL.

Oliveira, N., Cortez, P., and Areal, N. (2014). Automatic creation of stock market lexicons for sentiment analysis using stocktwits data. In *Proceedings of the 18th International Database Engineering & Applications Symposium (IDEAS 2014)*, page 115–123. ACM.

Oliveira, N., Cortez, P., and Areal, N. (2017). The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert System with Applications*, 73:125–144.

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. ELRA.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543. ACL.

Rao, D. and Ravichandran, D. (2009). Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, page 675–682, USA. ACL.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 1201–1211. ACL.

Tang, D., Wei, F., Qin, B., Zhou, M., and Liu, T. (2014a). Building large-scale Twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of the 25th International Conference on Computational Linguistics (Coling 2014)*, pages 172–182. ACL.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014b). Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1555–1565. ACL.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168.

Tsay, R. (2005). *Analysis of financial time series*. Wiley series in probability and statistics. Wiley-Interscience, second edition.

Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information System*, 21(4):315–346.

Vo, D.-T. and Zhang, Y. (2015). Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI 2015)*, page 1347–1353. AAAI Press.

Vo, D. T. and Zhang, Y. (2016). Don't count, predict! an automatic approach to learning sentiment lexicons for short text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 219–224. ACL.

Wang, C.-J., Tsai, M.-F., Liu, T., and Chang, C.-T. (2013). Financial sentiment analysis for risk prediction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP 2013)*, pages 802–808. ACL.

Zhang, L., Xiao, K., Zhu, H., Liu, C., Yang, J., and Jin, B. (2018). Caden: A context-aware deep embedding network for financial opinions mining. *2018 IEEE International Conference on Data Mining (ICDM 2018)*, pages 757–766.

Zhang, W. and Skiena, S. (2010). Trading strategies to exploit blog and news sentiment. In on Weblogs, F. I. C. and 2010), S. M. I., editors, *ICWSM*. AAAI Press.