Erasmus University Rotterdam
Erasmus School of Economics
Master Thesis MSc Data Science and Marketing Analytics

Comparative Analysis of Embedding Techniques for Sentiment Analysis in Finance

Chenyu Wang (541515)

Supervisor:	Eran Raviv
Second assessor:	
Date final version:	26th October 2023

Abstract

The research comprehensively compares various embedding techniques to assess their effectiveness in financial sentiment analysis. We begin by outlining financial English's linguistic features and highlighting the potential challenges for the embedding stage. In the following, we select representative embedding techniques from traditional word embedding (Word2Vec, GloVe, FastText), sense embedding (Sense2Vec), and contextualized embedding (BERT, ELMo, Embedding in GPT-2). Additionally, inspired by graph embedding, we also provide an innovative method to improve word embedding and contextuaAutoencoderdings with Autoencoder. Utilizing the embeddings, we proceed to train logistic regression and Support Vector Machine multiclass classification models for sentiment analysis based on Financial Phrase Bank. Then, we conduct intrinsic evaluation for embeddings by word similarity and extrinsic evaluation by multiple metrics. From the evaluation comparison, we conclude that the contextualized embedding technique, especially ELMo, outperforms other embedding techniques in most evaluation metrics. We also notice that Autoencoder-enhanced embedding has significant intrinsic improvement in terms of word similarity.

Keywords: word embedding, contextualized embedding, graph embedding, sense embedding, sentiment analysis, financial market, natural language processing

Contents

Chapter 1

Introduction

1.1 Background

Finance has long been recognized as an industry that relies on numbers, statistical analysis, and quantitative methods. However, in the past decade, people have seen a significant shift in this perspective as Machine Learning has begun to permeate various facets of the financial world. The integration of machine learning into finance has resulted in enhanced capabilities in risk prediction, quant trading, fraud detection, and credit scoring, among others. These techniques, leveraging Big Data and advanced algorithms, have enabled institutions to make more informed and efficient decisions, opening up new avenues for exploration and innovation in the field.

Amidst the myriad applications of machine learning in finance, sentiment analysis, an important branch of Natural Language Processing (NLP), stands out due to its unique role in interpreting the intangible aspect of financial markets: market sentiment. Market sentiment, reflecting stakeholders' collective attitudes and moods, has a long history of influencing market dynamics. A notable instance is the Tulip Mania in the Netherlands in the 17th century, a speculative bubble driven largely by market sentiment rather than fundamental value of tulip. Fast forward to the 21st century, the impact of market sentiment remains robust, for instance on stock returns and asset allocation (Malandri, Xing, Orsenigo, Vercellis & Cambria, 2018), monetary policy (Kashyap & Stein, 2023), and many other areas.

Given the importance of sentiment in finance (Baker & Wurgler, 2006), sentiment analysis, which involves extracting, interpreting, and classifying sentiment from textual data, has become a critical component in the Machine Learning toolkit for finance (Loughran & McDonald, 2011). From analyzing financial news (Tetlock, 2007), and social media posts, to earnings call transcripts (Davis, Piger & Sedor, 2012), sentiment analysis provides valuable insights into market dynamics (Tetlock, Saar-Tsechansky & Macskassy, 2008), investor behaviour (Frieder & Zittrain, 2007), and financial trends (Loughran & McDonald, 2016).

Central to sentiment analysis' performance is the technique of Embedding. Embedding, which maps objects to vectors of real numbers, lays a crucial foundation for text representation and interpretation in sentiment analysis and other downstream NLP tasks. Different embedding techniques can significantly influence the outcomes of sentiment analysis, including aspects such as accuracy, speed, precision, and recall, among others (Bakarov, 2018). This suggests that choosing an appropriate embedding technique is critical for optimizing sentiment analysis and

harnessing its full potential.

However, the exploration and understanding of how different embedding techniques perform when applied to financial sentiment analysis remain insufficient due to the sparsity of financial corpus(Malo, Sinha, Korhonen, Wallenius & Takala, 2014). This prompts the need for a comprehensive comparative study of various embedding methods in financial sentiment analysis.

1.2 Research Question and Thesis Structure

In this research, our objective is to undertake a comparative analysis of various embedding techniques, including word embedding, sense embedding, graph embedding and contextualized embedding. We will compare the efficiency and performance of various embedding techniques by implementing these methodologies within the context of the sentiment analysis based on Financial Phrase Bank(Malo et al., 2014). The intention of this analysis is to reveal unique attributes and potential limitations associated with each technique. Our ultimate goal is to formulate clear guidance regarding the situations and applications in which each word embedding technique may be optimally utilized, specifically within the scope of financial sentiment analysis. In conclusion, our central research question can be articulated as follows:

What are the comparative strengths and weaknesses of various embedding techniques when applied to financial sentiment analysis?

In order to address the central question of the research, the following structure has been adopted for this research. In Chapter 2, a comprehensive literature review is undertaken, focusing on prior comparative analyses of word embeddings, as well as linguistic features of English used in the finance realm. Chapter 3 provides an in-depth introduction to the Financial Phrase Bank. This includes a description of the dataset's composition, its structural characteristics, and a preliminary exploratory analysis of its data. Chapter 4 introduces the various embedding methodologies that we employ. These span from rudimentary techniques such as one-hot encoding to more advanced models based on autoencoders. Besides the embedding methods, evaluation metrics for assessing embedding techniques' performance are also presented. Additionally, this chapter introduces the machine learning approaches that we utilize for the training of the models as well. In Chapter 5, the resultant findings are analyzed and elucidated. A more detailed explanation for the difference in performance among embeddings is presented in Chapter 6. Furthermore, Chapter 6 concludes with a conclusive synthesis of the research findings. We find that, among all embedding techniques, contextualized embedding is believed to have the best performance in terms of both extrinsic and intrinsic evaluation metrics. Moreover, we also notice that Autoencoder can be used for optimizing traditional and contextualized embedding techniques' intrinsic evaluation task's performance. This chapter also provides recommendations concerning the potential optimal utilization of embeddings in the realm of financial sentiment analysis. The illustration of the research structure is shown in Figure 1.1.

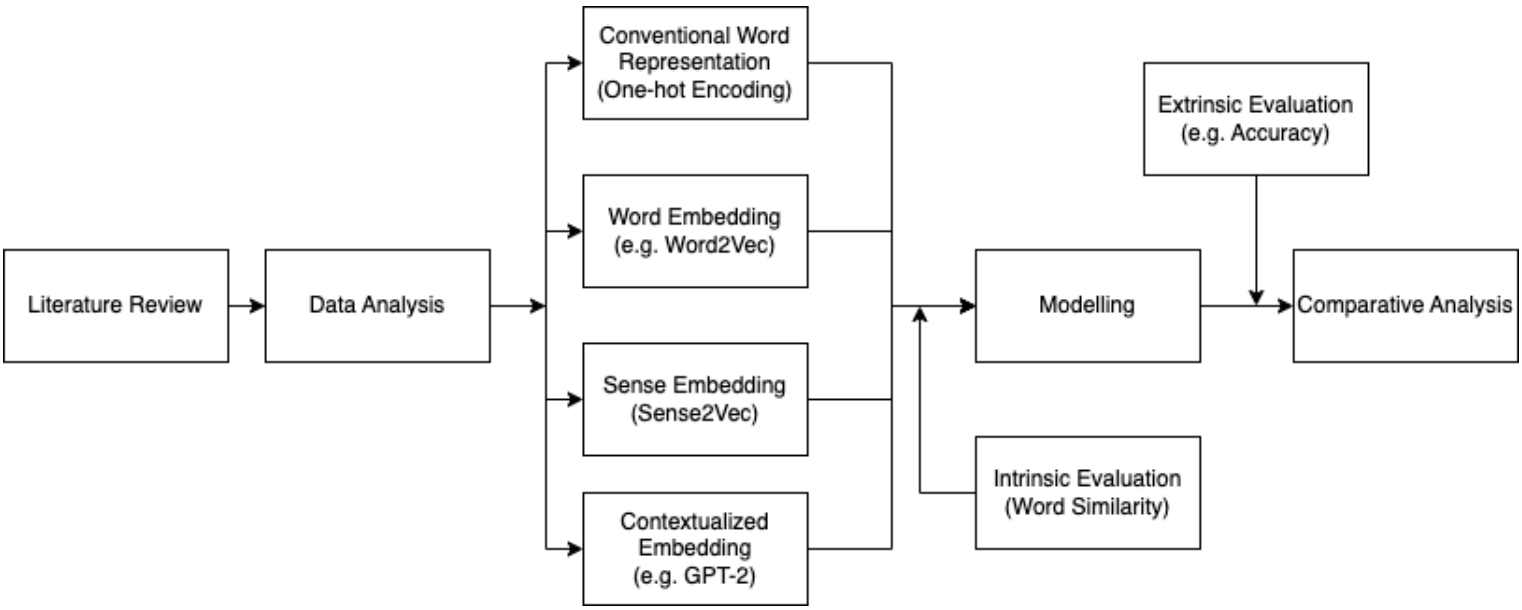

Figure 1.1: Research Design of Comparative Analysis of Embedding Techniques in Financial Sentiment Analysis

Chapter 2

Literature Review

In this chapter, we underscore the significance of selecting appropriate embedding techniques for financial sentiment analysis by considering the linguistic intricacies of financial English. Subsequently, we will examine extant comparative analyses of embedding techniques within the context of sentiment analysis and other analogous NLP tasks so that we can discover the existing research gap.

2.1 Linguistic Features of Financial English and Implications for Embedding Techniques Selection in Financial Sentiment Analysis

English is commonly known as the dominant language in finance realm. This dominance has also led to the development of a specialized form of English, namely, "Financial English". Financial English is characterized by its unique lexicon, syntactic structures, and semantic nuances. Nickerson (2005) highlights that this specialized language is replete with jargon, acronyms, and terms that might be ambiguous to the layperson but have precise meanings in financial contexts. Furthermore, the concise and unambiguous nature of financial reports and communications necessitates a specific linguistic structure, often favouring passive constructions and nominalizations (Dudley-Evans & St John, 1998).

These linguistic features become the challenge in NLP tasks of financial English after embedding starts to capture the semantic and syntactic relationships between words (Mikolov, Yih & Zweig, 2013). One of the main reasons is that the features can significantly influence the performance of embedding techniques. Given the specialized vocabulary and unique syntactic structures of Financial English, embeddings trained on general corpora might not capture these nuances effectively. This is supported by McEnery and Baker (2015), who state that domain-specific corpora are essential for pre-training that are sensitive to the linguistic peculiarities of specialized fields like finance. Moreover, the frequent use of jargon and technical terms in various realm may require embeddings that can capture domain-specific knowledge, which is highly rely on external resources such as pre-trained models (Rawte, Gupta & Zaki, 2020; Adhikari et al., 2023).

In the context of financial sentiment analysis, it is even more challenging to embed due to the

more nuanced difference between sentiment and often the understated language used in financial reports, comments and news. Loughran and McDonald (2011) note that traditional sentiment dictionaries often misclassify financial terms, leading to inaccurate sentiment scores. Based on all these challenges, the selection of embedding techniques plays an important role in achieving optimal financial sentiment analysis results (Dang, Moreno-García & De la Prieta, 2020).

2.2 Survey of Embedding Techniques Comparative Analysis

Due to the importance of embedding technique selection, many researchers work on evaluating and comparing the embedding techniques. A crucial decision in embedding selection is the choice between pre-trained embeddings and custom embeddings. Pre-trained embeddings, such as GloVe and FastText, encapsulate general language semantics and can be beneficial for tasks with limited data(Pennington, Socher & Manning, 2014), which is especially helpful in finance realm where available corpora is limited(Malo et al., 2014). On the other hand, some researchers state that custom embeddings which are trained on task-specific data can often outperform general embedding techniques in capturing domain-specific nuances. For instance, Araci (2019) creates a customed embedding model FinBERT based on BERT and finance corpora, and the model significantly outperforms general BERT in financial sentiment analysis.

Besides categorising embeddings into custom and pre-trained, Pilehvar and Camacho-Collados (2020) categorize embedding techniques into conventional embedding, word embedding (including sentence embedding and document embedding), sense embedding and contextualized embedding based on embeddings' target and training methods. Conventional embedding, such as one-hot encoding, paves the way for the following embedding techniques, but due to a lack of measuring word similarity, it is rarely used in nowadays' NLP tasks(Pilehvar & Camacho-Collados, 2020). Word2Vec and GloVe are considered as pioneers in vectorizing words in semantic space and start the era of embedding. However, they still have limitations, such as being computationally intensive and failing to handle out-of-vocabulary(OOV) words(Pennington et al., 2014; Mikolov, Chen, Corrado & Dean, 2013). Even though embedding techniques such as FastText innovatively generate embeddings for OOV words. But the embeddings are still suffer from the large number of dimensionality. After the introduction of the Transformer and biLSTM, embedding techniques such as BERT and ELMo become popular. Those techniques succeed in producing state-of-the-art context-dependent embedding that captures both syntactic and semantic information of words. But all these methods require computational resources(Peters et al., 2018; Devlin, Chang, Lee & Toutanova, 2018) and the generative model may not always produce embedding that is optimal for domain-specific tasks(Radford et al., 2019). Autoencoder-based models such as AutoExtend is also receiving more and more attention these days. The models can capture non-linear relationships in the data and enhance the quality of embeddings. However, the performance is highly dependent on tuning and may lead to overfit(Makhzani, Shlens, Jaitly, Goodfellow & Frey, 2015).

These previous researches help us understand the strengths and weaknesses of different embedding techniques. The choice of embedding technique should be informed by the specific requirements of the NLP task at hand. While some embeddings excel in capturing semantic relationships, others might be more suited for domain-specific tasks that require an understanding

of specific knowledge. In that case, evaluation methods for word embedding are crucial.

Schnabel, Labutov, Mimno and Joachims (2015) summarizes embedding evaluation metrics into two classes: extrinsic evaluation and intrinsic evaluation. Researchers can evaluate embedding based on the technique itself by comparing word similarity or evaluating the performance of downstream NLP tasks like sentiment analysis. Bakarov (2018) is seminal for providing a comprehensive survey on both intrinsic and extrinsic evaluation methods of embeddings. The author identifies several major challenges in evaluating word embeddings. These include the inherent obscureness of semantics, the absence of proper training data for evaluation, a lack of correlation between task-dependent intrinsic and extrinsic evaluation methods, the absence of significance tests, and the hubness problem. The author states that due to various limitations, no single evaluation metric is superior. Evaluation should be comprehensive and task-specific.

2.3 Research Gap

The existing literature extensively covers embedding techniques and their respective applications. However, a discernible void exists regarding comprehensive comparative analysis within the finance domain, particularly concerning financial sentiment analysis. Owing to the distinct linguistic features inherent in financial texts, an in-depth understanding of the efficiency and performance of various embeddings is paramount. Thus, an urgent demand emerges for research that probes into the comparative analysis of embedding techniques specifically tailored for financial sentiment analysis.

Furthermore, current academic discourse on embedding comparisons mainly centres around traditional word embedding methodologies such as GloVe and Word2Vec or contemporary techniques like ELMo and BERT. Due to the popularity of Transformer-based architectures in the field, models rooted in Autoencoders and those incorporating enhancements through Autoencoders are conspicuously underrepresented. In our research, we shall also go beyond recognized embedding strategies and explore the potential contributions of Autoencoders to embedding.

Chapter 3

Data

In this chapter, we are going to introduce the database we will use for research, which is Financial Phrase Bank. We will first delve into the origin and structure of the database, then we will conduct a series of in-depth analyses of our database to gain a more profound understanding of our data.

3.1 Data Gathering and Organization

As mentioned before, to advance the comparison of word embedding techniques in financial sentiment analysis, this study employs a dataset known as the Financial Phrase Bank from Malo et al. (2014), which was developed with support from the Emil Aaltonen Foundation and the Academy of Finland in 2013. This dataset aggregates 4840 sentences and all the sentences come from finance-related English news, comments and research reports of listed companies in OMX Helsinki. The dataset counters the scarcity of quality annotated data for model training in the financial domain.

According to Malo et al. (2014), all sentences in the Financial Phrase Bank are annotated by finance professionals, which means individual subjective opinions highly affect the annotation outcome. The authors separate sentences into four categories based on the strength of majority agreement. For the purpose of our research on comparing embedding techniques, we would like to use sentences with 100% agreement rate. Our final selected dataset consists of 2264 rows of sentences with the texts themselves and corresponding sentiment annotation.

3.2 Data Pre-processing

In the financial phrase bank, each sentence has been categorized into one of three distinctive sentiment classes: positive, negative, or neutral. As illustrated in Figure 3.1, the dataset comprises a total of 303 sentences that have been annotated with negative sentiment and 570 sentences as positive sentiment. In contrast, a significantly larger portion, amounting to 1391 sentences, has been identified as neutral. This classification was determined by the inherent sentiment conveyed within the content of each sentence. Furthermore, as a part of our analytical approach in the study, we transform sentiment to numeric value so that the sentiment score can fit in certain embedding techniques and the following modelling stage. In our dataset, neural sentiment is

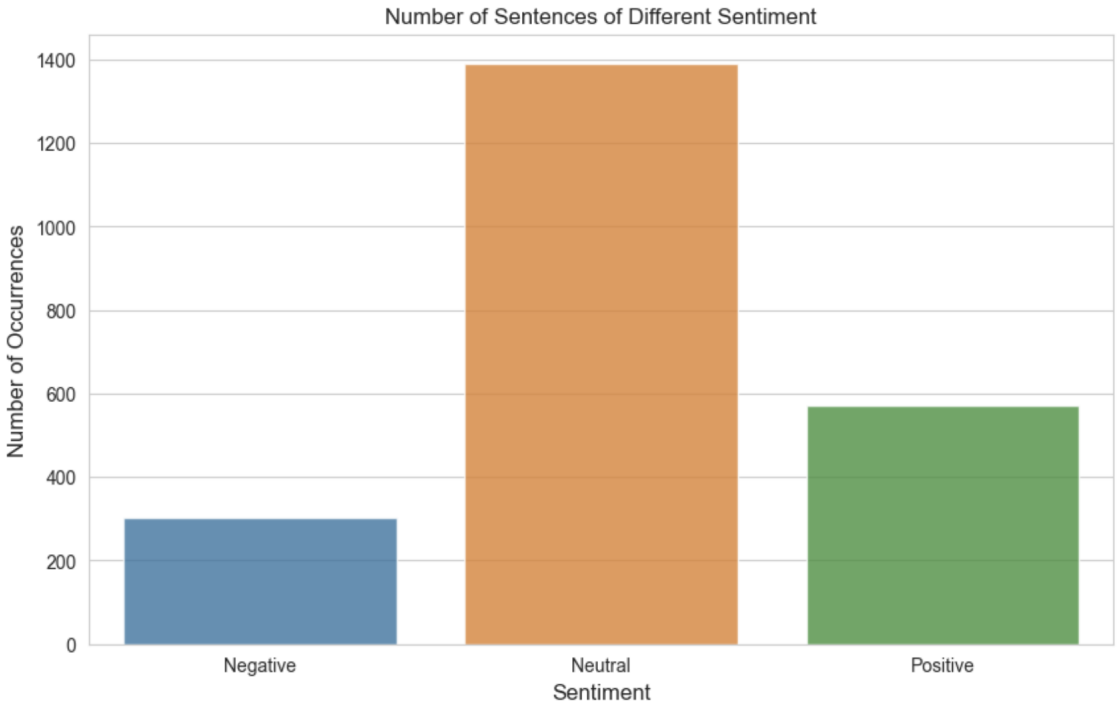

Figure 3.1: Distribution of sentiment classes in the selected Financial Phrase Bank dataset. The Y-axis shows the number of occurrences of a certain sentiment in the dataset. The X-axis shows the name of sentiment, from left to right, they are "negative", "neutral" and "positive".

labelled as 0, positive as 1 and negative as -1. We provide the initial five data samples in Table 3.1 to help understand the data structures.

Table 3.1: The first five samples of the selected Financial Phrase Bank dataset

Index	Sentence	Sentiment	Value
0	According to Gran , the company has no plans t...	Neutral	0
1	Technopolis plans to develop in stages an area...	Neutral	0
2	The international electronic industry company ...	Negative	-1
3	With the new production plant the company woul...	Positive	1
4	According to the company 's updated strategy f...	Positive	1

3.3 Explanatory Data Analysis

In order to gain more insight into the dataset and detect potential challenges in the following stages of the study, we conduct a series of explanatory data analysis to highlight its statistical characteristics. In this section, we will present the results of the LDA, TF-IDF values, and WordCloud and the findings from the results.

Average Term Frequency-Inverse Document Frequency Value Rank

In order to gain a comprehensive understanding of the frequency of related words in every sentence, we first determined the TF-IDF values of each word. TF-IDF stands for Term Frequency-

Inverse Document Frequency, it is a metric that could be used for measuring the frequencies of words in a large corpus. It measures the importance of a term in a document relative to its importance across all documents in the corpus. Mathematically, TF-IDF values are calculated by equations 3.1 to 3.3.

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ occurs in document } d}{\text{Total number of terms in document } d} \tag{3.1}$$

$$\text{IDF}(t, D) = \log \frac{\text{Total number of documents in corpus } D}{\text{Number of documents containing term } t} \tag{3.2}$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{3.3}$$

The top 20 most frequent words are presented in Figure 3.2. In Financial Phrase Bank, the 5 most frequent words are "eur", "profit", "net", "operating", and "sales". The prominence of "eur" underscores the dataset's European or specifically Finnish context, given that the euro is the official currency of Finland. Terms like "profit", "net", "operating", and "sales" are foundational in financial reporting and analysis, indicating that the dataset contains a wealth of information on companies' financial performances and operations. The outcome confirms the description provided by the authors, namely, the comments primarily focus on the financial metric in European or Finnish Economics.

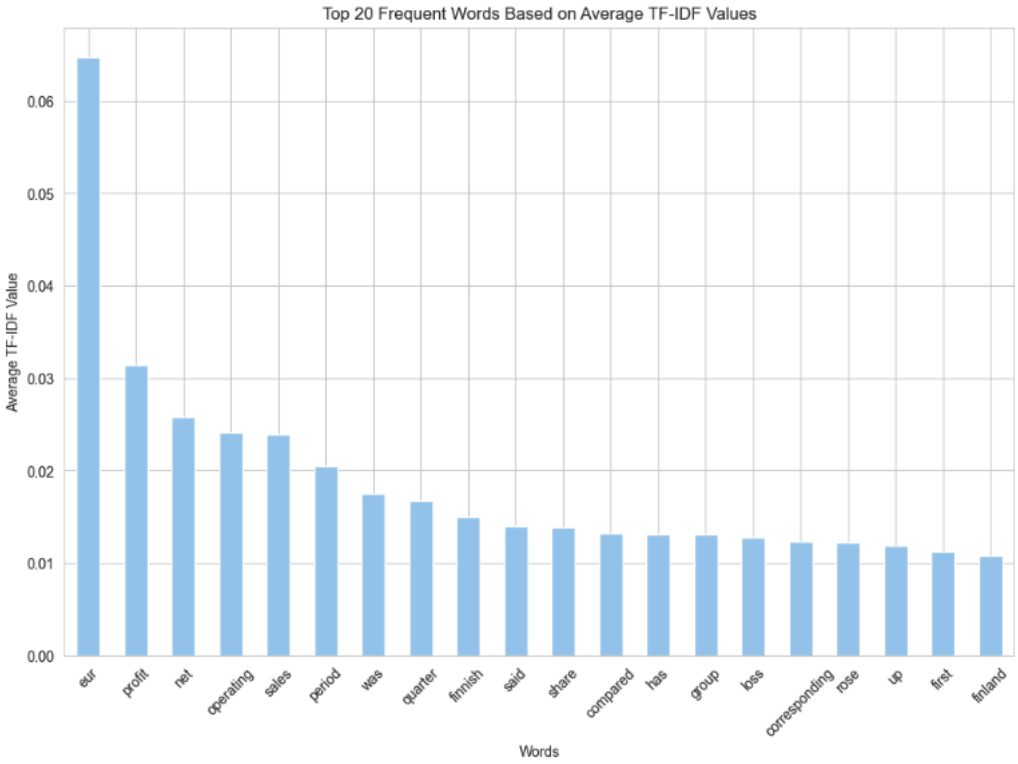

Figure 3.2: Top 20 frequent words based on average TF-IDF. The X-axis displays the details of the words and the height of each bar shows the average TF-IDF value of the corresponding word.

Latent Dirichlet Allocation Result

Latent Dirichlet Allocation (LDA) is a probabilistic model that presents the underlying thematic structure of all documents. By assigning mixtures of topics to documents and words to topics, LDA allows for the interpretation of text corpora in terms of a few key topics. LDA outcome provides a quick overview of the main themes present in the Financial Phrase Bank.

In the context of the Financial Phrase Bank, by setting topic numbers to three, the outcome is shown in Figure 3.3, :

Topic 1 is about market dynamics and company performance, with keywords like "share," "sales," "market," and "company." This topic likely captures discussions about market shares, sales figures, and overall company standings in the Finnish market.

Topic 2 focuses on financial metrics with terms such as "EUR," "mn" (million), "profit," "sales," and "period." This suggests discussions related to financial reporting, earnings, and specific monetary values indicate quarterly reports or financial summaries.

Topic 3 has a regional focus, with the term "Finnish" being prominent alongside "mln," "euro," and "said." This topic might capture news or reports specifically related to Finnish companies or the Finnish economy, which is again in line with the dataset description above.

Instead of going through each sentence, we can get a sense of the dataset's content by looking at these topics. Furthermore, for sentiment analysis, these topics can be crucial. Knowing the prevalent themes can help in understanding the context in which sentiments are expressed. For instance, sentiments in financial reporting (topic 2) might be more objective and data-driven, while sentiments in market dynamics (topic 1) might be more speculative or opinion-based.

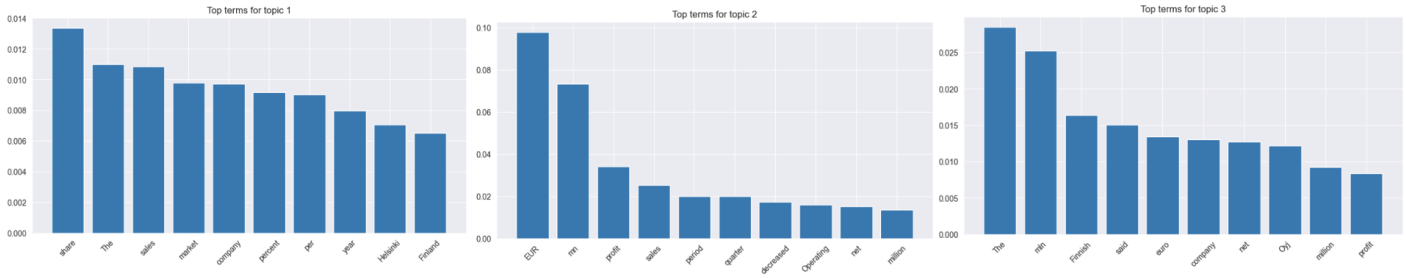

Figure 3.3: Outcome of LDA analysis. The X-axis displays the terms. The Y-axis represents probabilities associated with each term in the specific topic. The height of each bar indicates the relevance of a term to the corresponding topic. From left to right, they are topic 1,2 and 3.

Word Cloud

Word Cloud, firstly proposed by Bausch and Bumgardner (2006), is a visual representation of words in a corpus that could indicate the frequencies and importance of words. It is worth noticing that, in the following visualization, we remove all articles, prepositions and ordinal numbers.

Detailed Word Cloud visualization can be found in Appendix A. Figure A.1 shows the word cloud for comments with negative sentiment. The word cloud is dominated by terms describing

decline such as "down" and some financial terms that may have a potential negative impact on market sentiment, such as "operating loss". In contrast, Figure A.2 reflect a more optimistic tone by terms such as "operating profit". Figure A.3 shows that neutral comments are mostly characterized by terms that are not directly related to financial operations, such as "service", "share", and "business". We propose that this may be because neutral comments are mainly statements of facts.

Dataset Characteristics Summary

Based on the TF-IDF and basic information of the dataset, we notice that the sentiment of the dataset is not balanced. An imbalanced dataset may lead to misclassification in the later modelling and test stage. However, we believe it reflects a scenario closer to reality. As mentioned before, financial English is considered to be understated and more likely to convey neutral sentiment because most professionals tend to give out neutral advice and comments regarding financial markets. In that case, we would like not to take action against this challenge.

According to LDA results, we can divide the topics into financial metrics, market dynamics, and Finnish economics. Even though the whole corpora is under the theme of the finance industry in Finland, however, internal topics among sentences are various. Diversity of topics may lead to many challenges in embedding. Firstly, diverse topics may cause some topics to be underrepresented due to the sparsity of the corpora. In such cases, we believe contextualized embedding models are more likely to outperform. Secondly, there is a higher chance of encountering out-of-vocabulary words, even for using the pre-trained models. Last but not least, some evaluation metrics, such as analogy tasks, may not be comprehensive enough to measure quality across topics.

From the Word Cloud, we can notice two challenges we may face in the following sentiment analysis. In the first place, we see a significant overlapping of words between different sentiments. For instance, common financial terms such as "operating profit" are widely used in both positive and negative comments. Moreover, some words may refer to different sentiment, and the classification of these words highly depend on contexts, such as "loss". These may potentially affect the performance of embeddings based on their capability of disambiguation. In that case, taking senses and contexts into account is necessary for better word representations.

In the following chapters, we will combine the features of the data we learn from the analysis to model selection, result explanation and discussions.

Chapter 4

Method

In this chapter, we will introduce the involved embedding techniques and machine learning method we used for the following comparative analysis. Generally speaking, the embedding techniques we use can be categorized into four classes based on the framework in Pilehvar and Camacho-Collados (2020): word embedding, sense embedding, graph embedding and contextualized embedding.

Descriptions pertaining to the implementation of each embedding technique are presented in Table B.1 and Table B.2 in Appendix B. This table clarifies whether certain embedding technique is pre-trained or customed and provides an introduction to the pre-trained models we utilized.

This research covers selected models that are popular in each class so that it can show how different categories of embedding affect the performance of sentiment analysis and also compare across embedding techniques. For each technique, we will introduce the architecture and basic theory. For the purpose of comparison, we will introduce evaluation metrics after the embedding techniques. In the end, we will also give a brief introduction to the Support Vector Machine and Logistic Regression we use for training models.

4.1 One-hot encoding

Before the 1990s, count-based models for constructing discrete vector space are dominant in word representation. One-hot encoding is the most basic Term-document count-based word representation. The technique represents each word in the corpus as a unique vector, and the vector has a dimensionality that equals to the size of the vocabularies. As shown in the example in Figure 4.1, in a sentence with four words, one-hot encoding representation constructs a 4 x 4 matrix. Such representation technique has salient limitations. First of all, the way it constructs vector lead to high dimensionality, which makes it highly storage-sensitive when dealing with large datasets in the era of Big Data. Furthermore, such representation has no measurement of a word's semantics. One-hot encoding is a traditional and simple count-based word representation, but it lays the foundation for the development of other embedding techniques and paves the way for more complex and efficient models. Basically, all successor techniques inherit the idea of vectorizing words.

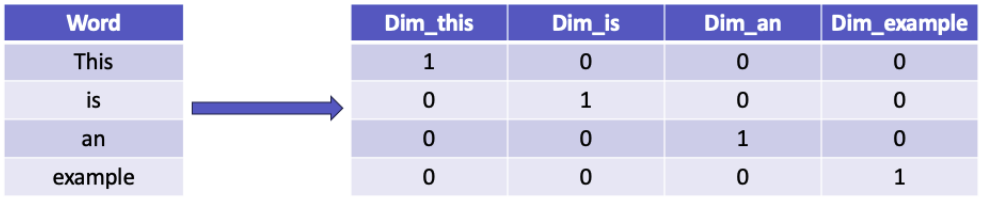

Figure 4.1: One-hot encoding illustration. The left part is text input and the right part illustrate how the word are encoded to vectors.

4.2 Word Embedding

Salton, Wong and Yang (1975) proposes the Vector Space Model (VSM) to present a solution to the limitations of one-hot encoding, such as unreasonable variable size and no capability of representing words' semantics and syntaxes. In the context of NLP, it represents objects such as words, sentences or documents in a multi-dimensional space called semantic space. Unlike one-hot encoding, semantic space is distributed and continuous. VSM is built upon distributional hypothesis (Firth, 1957), meaning words' semantic similarities can be measured by their statistical distance. For instance, in our dataset, "profit" and "sales" frequently co-occur in comments because they usually both appear in the context of the financial performance of companies' operations. In this section, we will introduce three traditional and powerful word embedding techniques based on VSM theory.

4.2.1 Word2Vec

Word2Vec, firstly proposed in Mikolov, Chen et al. (2013), is seen as the technique that start the era of neural network application in word representation. It popularised word embedding in various NLP tasks, including sentiment analysis. The general idea under Word2Vec is to represent words as vectors in multi-dimensional semantic space and, as a VSM, words that appear in similar contexts have lower statistical distance. Mikolov, Chen et al. (2013) proposed two different Word2Vec models, namely, Continuous Bag-of-Words and Skip-Gram. We will utilize both models for our analysis.

Countinuous Bag of Words (CBOW)

The CBOW model predicts the target word by its surrounding words. The model aims at minimizing the negative value of the probability of occurrence of a target word given the context. Mathematically, the loss function can be represented as:

$$\mathcal{L}(\theta) = -\log(p(w_t|W_t)) \tag{4.1}$$

where w_t represents the target word and W_t is the matrix constructed by the surrounding words vectors, namely, $W_t = w_{t-n}, ..., w_t, w_{t+n}$. The architecture of CBOW is shown in Figure 4.2. The Embedding process of the CBOW model has an input layer, a hidden layer and an output layer. The input layer consists of the one-hot encoded surrounding words. After feeding the input to the hidden layer, the word vectors are multiplied by the weight matrix and

construct a dense word vector. Based on the dense vector, the output layer predicts the target word by minimizing the loss function, namely, finding the word that has the highest probability of occurrence.

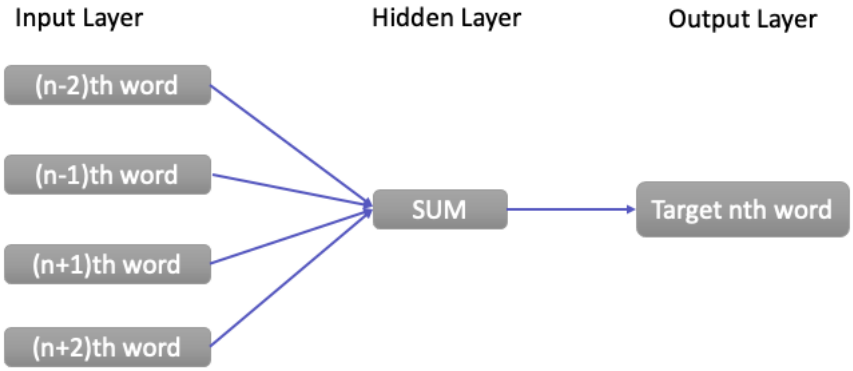

Figure 4.2: Architecture of Continuous Bag of Words. The figure illustrates an instance of CBOW: Two words before the target word and two words after the target word are fed into the input layer. After summation, the target nth word is predicted.

Skip-Gram Model

Skip-Gram model has a reverse architecture of CBOW model. As shown in Figure 4.3, its input layer is the target word, and the output layer has as many neurons as the words in the vocabulary. It provides the probability distribution of all words in the vocabulary, and the contextual words are predicted based on the likelihood of co-occurrence with the target word as well. The objective function is shown as:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t) \tag{4.2}$$

where T is the size of the vocabulary, and the inner summation is over the surrounding words of the target word w_t.

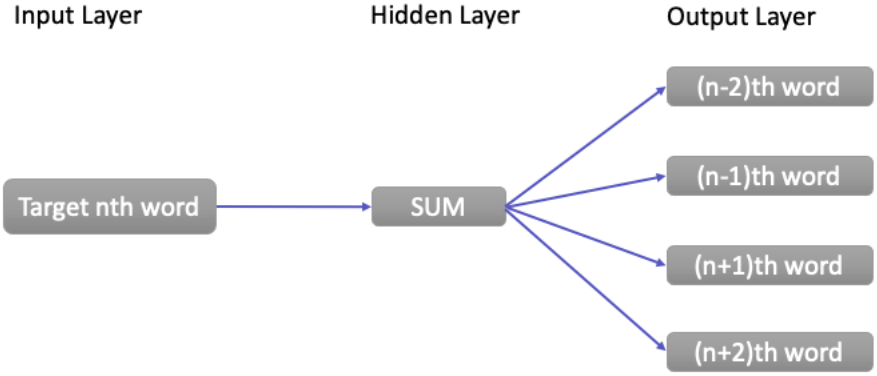

Figure 4.3: Architecture of Skip-Gram model

4.2.2 GloVe

Another popular word embedding technique we use in our analysis, GloVe, is first proposed in Pennington et al. (2014). Similar to Word2Vec, it is also a technique for representing words in semantic space. The main idea of GloVe is using ratios of word co-occurrence probabilities. GloVe aims to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Given a co-occurrence matrix X, where X_{ij} represents frequencies of co-occurrence between word i and word j. The mathematical representation is shown as follows:

$$\mathcal{L} = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2 \tag{4.3}$$

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{4.4}$$

where f is a weight function that assigns weights based on frequencies of co-occurrences. w_i and w_j are vector representations for word i and word j. b_i and b_j are bias terms. Unlike the conventional word representations we mentioned in the last section, GloVe captures the semantics of words, which are crucial for our sentiment analysis, where semantics plays an important role. Furthermore, as a pre-trained model, we can take advantage of the massive corpora the model is trained on.

4.2.3 FastText

Word2Vec and GloVe are both considered predictive models since they both involve predicting target words in a context based on neural networks. However, if the word is out-of-vocabulary (OOV), the previous embedding technique may not be efficient. In that case, this may lead to misunderstanding if the OOV word is the keyword of the context or document. Solutions to such challenges are crucial in financial sentiment analysis. One of the most important reasons is that language in the finance area is dynamic. Every new emerging industry, company, event or technology advancement may lead to new words or phrases that may not be contained in the pre-trained embedding model's vocabulary. Especially in our case, as shown in our LDA results, diverse topics lead to the possible existence of domain-specific keywords.

To solve this problem, a series of embedding techniques, which is known as character embedding, is developed. FastText is one of them and has better performance in handling OOV words.

FastText is an open-source library developed by Facebook (Bojanowski, Grave, Joulin & Mikolov, 2017). The main idea behind this model is breaking words in a corpus into a group of semantically meaningless character n-grams. For instance, "finance" can be represented by "fin", "nan", "nce", and so on. Such n-gram representation can partly solve the OOV word challenge. But we should not ignore the fact that two different words may have similar n-gram constituents, for example, "capital" and "capitol".

4.3 Sense Embedding

Based on the previous introduction to word embedding and conventional word representation. We can easily notice that they are under the objective of representing words as a single point in a static semantic space. Such representations lead to Meaning Conflation Deficiency, which means that it is not possible to unambiguous lexical meaning of a single word (Schütze, 1998). The deficiency can significantly hamper the performance of the downstream NLP task. This is crucial in the finance area as well. For instance, the word "bond" is commonly used to refer to debt security, while it can also be used to describe relationships between target companies and stakeholders. In that case, conflating the meaning of "bond" into one single point will lower the accuracy of sentiment analysis. There are a lot of similar situations, such as "interest", "bank", and so on, so it is crucial to take the sense of financial words into account when embedding (Li & Jurafsky, 2015). A series of embedding techniques, which is known as "sense embedding", has been developed to alleviate the meaning conflation deficiency. In the next section, we will introduce Sense2Vec, a sense embedding technique we will use in our research.

4.3.1 Sense2Vec

Sense2Vec, proposed in Trask, Michalak and Liu (2015), addresses the challenge of word sense disambiguation in previous word embeddings. Traditional word representations often provide a single representation for each word, but Sense2Vec takes the polysemous nature of financial English words and phrases into account. Sense2Vec models multiple embeddings for each word based on supervised disambiguation. This means that a word with multiple senses will have distinct embeddings for each sense. When integrated into downstream NLP tasks such as sentiment analysis, Sense2Vec allows these models to select the most appropriate sense-disambiguated embedding for a word based on its context. This selection process enhances both the accuracy and efficiency of the following model. Beyond just disambiguating between completely different senses of a word, Sense2Vec is also capable of distinguishing between nuanced senses, providing a richer representation of language.

As illustrated in Figure 4.4, in the example sentence "Banks at the bank", the word "bank" is simply represented by a single vector in traditional semantic space, such as Word2Vec. In contrast, Sense2Vec provides multiple embeddings for a single word, and we can distinguish the verb "bank" and noun "bank" through sense embedding.

4.4 Contextualized Embedding

Usually, Word2Vec, GloVe and FastText are considered as static embedding techniques. Namely, the representation of words in the corpus is fixed. In contrast, Sense2Vec is considered a dynamic embedding. In the following, we will introduce another type of dynamic embedding technique, which is commonly called contextualized embedding. Contextualized Embedding represents words based on context and gives more representations for a single word than Sense embedding. Take the previous "bank" as an example, Word2Vec and GloVe have static single representations for the word, and sense embedding provides two embeddings based on the word's sense. Dynamic contextualized embedding, however, provides different vectors for "bank" based on its

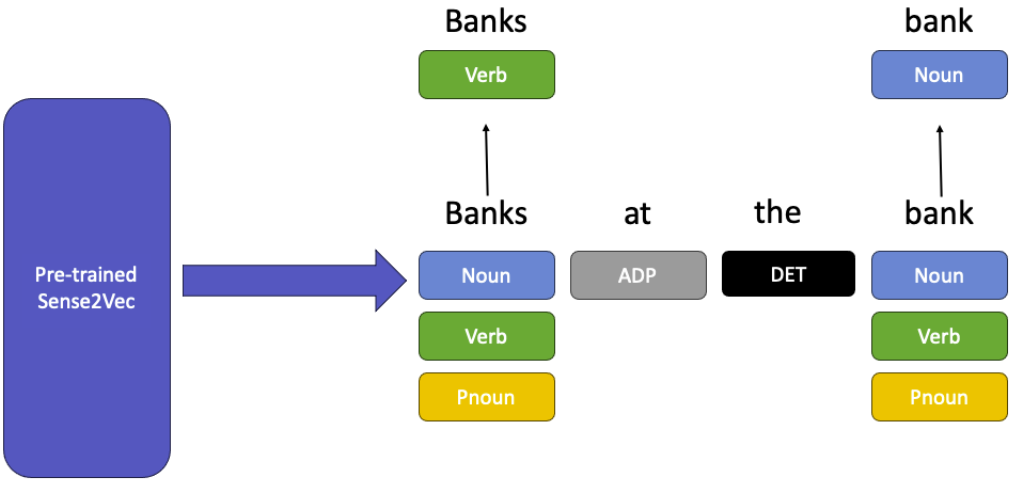

Figure 4.4: Sense2Vec illustration. In the instance, Sense2Vec pre-trained model has three embeddings for "bank" based on its sense, namely, noun, verb and pronoun. In the example sentence "Banks at the bank", the two "bank" have different senses and, in that case, have different embeddings.

surrounding context. In this section, LSTM-based model ELMo and Tranformer-based model GPT-2 and BERT will be introduced.

4.4.1 ELMo

Embeddings from Language Model (ELMo), proposed in Peters et al. (2018), takes the whole sentence as input and are also considered as a type of sentence embedding. ELMo is essentially a contextualized embedding model that captures the features of both syntax and semantics of sentences while capturing the features across the context.

ELMo is based on a bidirectional LSTM (biLSTM). Figure 4.5 depicts the architecture of ELMo. The text is firstly converted to vectors based on a conventional neural network (CNN), and the word vectors are then regarded as the input of the biLSTM. The biLSTM consists of a forward pass and a backward pass. The forward pass layer reads the sentence from left to right, in other words, it reads words before the target word, while the backward reads the sentence from right to left or reads the words after the target. The information from the two layers will be combined and form an intermediate vector to represent the target, and then the intermediate word vectors will be fed into the next layer. To obtain the final embedding, ELMo aims at minimizing the loss function, which is the sum of the forward and backward log-likelihoods for a given sequence of tokens. Mathematically, the formula is shown as below:

$$\mathcal{L} = -\left(\sum_{t=1}^{T} \log P\left(w_t | w_1, w_2, \ldots, w_{t-1}\right) + \log P\left(w_t | w_{t+1}, w_{t+2}, \ldots, w_T\right)\right) \tag{4.5}$$

where w_t is is the token at position t in the sequence. T is the total number of tokens in sequence. The first term in the summation is the forward log-likelihood, and the second is the backward log-likelihood.

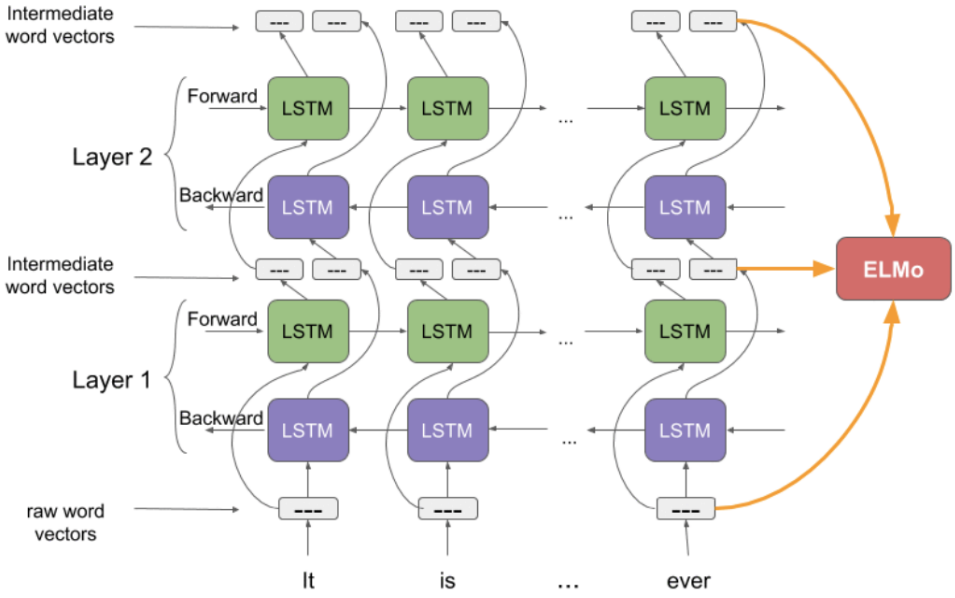

Figure 4.5: Diagrammatic representation of ELMo (Verma & Sharma, 2020)

4.4.2 Basics of BERT and GPT-2: Transformer

The Transformer architecture represents a significant shift away from traditional RNN and CNN-based approaches in NLP. It is first introduced by Vaswani et al. (2017). Instead of relying on sequential processing or local convolutions, the Transformer model utilizes attention mechanisms to draw global dependencies between input and output, enabling parallel processing and catering to long-range dependencies.

The core innovation in the Transformer architecture is the "self-attention" mechanism. This allows the model to weigh the significance of different parts of the input data, permitting each word in a sequence to focus on other relevant words in the same sequence, disregarding their distance (Vaswani et al., 2017).

In detail, for each input object, which is the context in our case, the model computes three vectors, Q, K and V, which stand for queries, keys and values. The three vectors are computed as

$$Q = XW_Q \tag{4.6}$$

$$K = XW_K \tag{4.7}$$

$$V = XW_V \tag{4.8}$$

where W_Q, W_K, W_V are weight metrics, X is the matrix of the input objects. At the beginning, the weights are randomly initialized, and during the forward propagation, input embeddings are multiplied with weight matrices to produce queries, keys and values. The loss value calculated from the forward stage will be then backwards propagation to calculate gradients for all parameters and then update the parameters. Through the iterative process, the model will converge at the optimal values.

Given an object, its attention score with any other objects in the sequence is calculated as

$$\text{Score}(Q, K) = QK^T \tag{4.9}$$

For stabilizing the gradients during the training, scaling is needed, which is

$$\text{Scaled Score} = \frac{\text{Score}(Q, K)}{\sqrt{d_k}} \tag{4.10}$$

where d_k is the dimensionality of the key vectors.

$$\text{Attention Weights} = \text{softmax}(\text{Scaled Score}) \tag{4.11}$$

$$\text{Output} = \text{Attention Weights}V \tag{4.12}$$

Based on softmax normalization and the weighted sum of values, the output is predicted.

Figure 4.6 depicts the Transformer model's core components, with two main sections, encoders and decoders, which are for processing input and generating output, respectively. Both the encoder and decoder consist of several identical layers stacked on top of one another, as represented by the "Nx" notation. Each layer in the encoder contains two primary components: Multi-Head Attention and Feed Forward Neural Network. The decoder has an additional Masked Multi-Head Attention component.

Transformers revolutionized NLP tasks by setting new state-of-the-art performances across various benchmarks. This includes tasks like machine translation, question answering, and named entity recognition, among others. Unlike RNNs, which process sequences token-by-token, Transformers handle entire sequences at once, offering inherent parallelism. This makes them scalable and well-suited for modern hardware accelerators.

Traditional models like LSTMs and GRUs often struggle with very long sequences due to the vanishing gradient problem. Transformers, with their attention mechanisms, can handle long-range dependencies with ease. Nowadays, due to its architecture and performance, Transformer-based models are widely used for embedding.

4.4.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is developed by researchers at Google (Devlin et al., 2018). Word embedding techniques and sense embedding techniques we mentioned before all operate in a unidirectional manner, either forward or backwards. But BERT is designed to consider context from both directions, allowing it to capture a more comprehensive understanding of each word in a sentence. As mentioned at the beginning of this section, BERT leverages the Transformer architecture, utilizing self-attention mechanisms to assess and assign weights to words in a sentence related to a specific word.

BERT's pre-training involves two main tasks. The first one is Masked Language Model. In this task, a certain percentage of the input data is randomly selected to be hidden. The model then predicts the original masked words based on their surrounding contexts. The loss function for this task is the negative log-likelihood of the masked words, which means the model is penalized based on how far its predictions are from the actual words. The formula of the loss

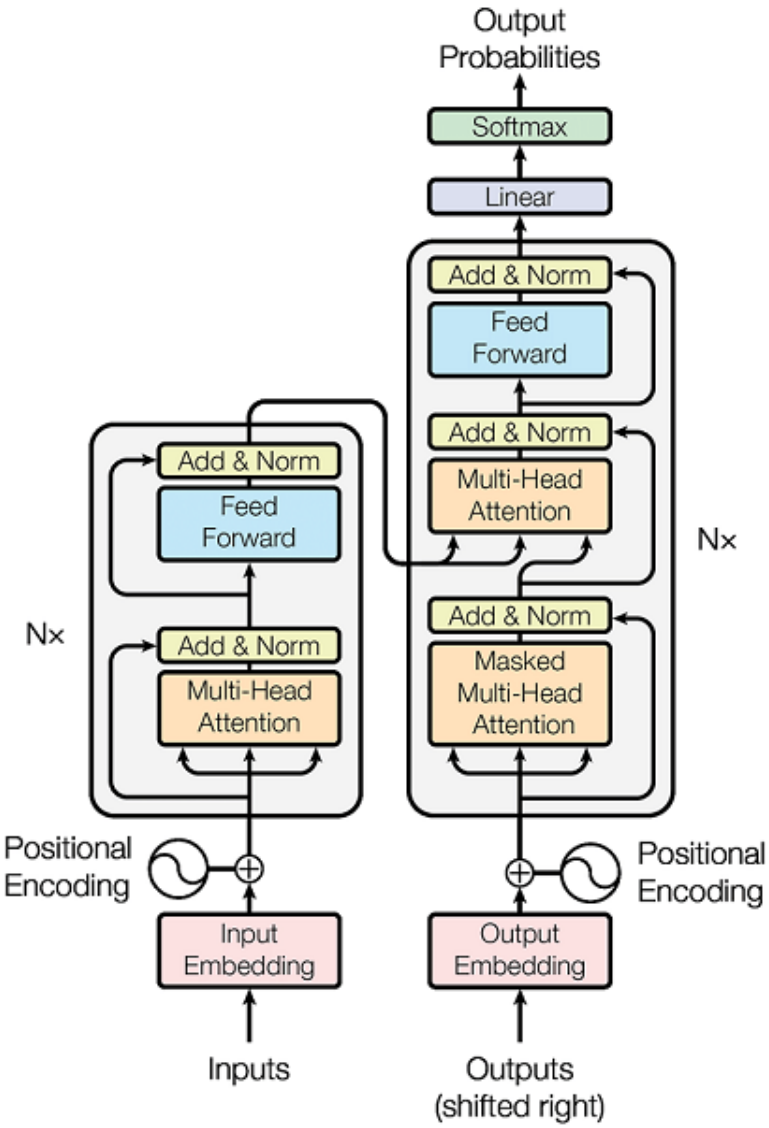

Figure 4.6: Architecture of Transformer (Vaswani et al., 2017)

function is shown as below:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i=1}^{N} \log P(w_i|\text{context}) \tag{4.13}$$

The second task is Next Sentence Prediction. For this task, the model is given pairs of sentences and predict whether the second sentence in the pair follows the first sentence in the original text. The loss for this task is the binary classification loss between the predicted and actual labels, which is:

$$\mathcal{L}_{\text{NSP}} = -\sum_{j=1}^{M} y_j \log(\hat{y}_j) + (1 - y_j) \log(1 - \hat{y}_j) \tag{4.14}$$

The complete loss function of BERT is the sum of the two losses above, which is:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}} \tag{4.15}$$

4.4.4 Embedding in GPT-2

GPT-2, which stands for "Generative Pre-trained Transformer 2", is developed by OpenAI in Radford et al. (2019). It is built upon the Transformer architecture as well. Unlike the typical transformer model, GPT-2 discards the encoder part and directly uses sentences as input in decoder. The objective function of GPT-2 is to maximize the likelihood of a word sequence based on its preceding words. Mathematically, this can be represented as:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \log P(w_t|w_1, w_2, \ldots, w_{t-1}; \theta) \tag{4.16}$$

where w_t is the word at time step t, T is the total number of words, and θ represents the model parameters.

The gradients required for optimization are derived using backpropagation, with the gradient of the loss with respect to the model parameters given by:

$$\nabla_\theta \mathcal{L}(\theta) \tag{4.17}$$

It is worth noticing that GPT-2 is not an embedding technique in essence. But it is feasible to make use of the embedded sentences in the GPT-2 hidden state. In practice, we first tokenized the sentence using the GPT-2 tokenizer and then passed the tokenized sentence through the GPT-2 model to obtain the hidden states for each token. Averaged the hidden states of all tokens in the last hidden state of the last layer to get an embedding for the entire sentence, and these embeddings are what we use as input in the downstream sentiment analysis task. The last layer is known to capture high-level semantic information, making it ideal for understanding the intricate nuances of financial language.

The choice of contextualized embedding techniques for sentiment analysis of the Financial Phrase Bank is backed by several compelling reasons. Firstly, these embeddings are inherently contextual. In the realm of finance, where a word's sentiment can pivot based on its context,

this property is invaluable. For instance, as mentioned in Chapter 3, the word "loss" might have a neutral sentiment in a statement about "loss coverage ratio" but could be negative in the context of "significant quarterly loss." Secondly, the comprehensive training data of contextualized embedding ensures a foundational understanding of language. When combined with fine-tuning, it makes the model adept at discerning sentiments in intricate financial statements. Lastly, the capability to fine-tune models means that the embeddings can be tailored to capture the unique sentiment nuances present in specialized financial corpora like the Financial Phrase Bank.

4.5 Graph Embedding

Autoencoder-based models are commonly used for graph embedding. Graph is a widespread data structure which is also commonly used for representing semantic networks and for semantic modelling in language. In the scenario of sentiment analysis, where data are often in the form of texts, graph embedding is also helpful for exploring the relations between texts. Kaneko and Bollegala (2020) state that pre-trained embedding can be improved by converting word vectors to graph structures and through Autoencoder-based node embedding and relation embedding techniques, and it is possible to represent information from texts in graphs. Li and Jurafsky (2015) state that such autoencoder-enhanced embedding model can generate new well-represented word vectors apart from pre-trained embedding models. In this section, we will introduce this potential method of improving embedding techniques.

4.5.1 Autoencoder-enhanced Embedding

Autoencoder outperforms at compressing high-dimensional graph data into more compact, lower-dimensional representations. This capability is invaluable for diminishing the computational complexity in downstream tasks. Additionally, it facilitates the visualization of graphs in a low-dimensional space.

By capturing the inherent structure and patterns present in the graph, Autoencoder can derive meaningful representations of graph nodes. The training process, which involves reconstructing the input graph, enables the hidden layers of the Autoencoder to discern vital features and inter-node relationships.

One of the valuable features of Autoencoder is its ability to be trained in an unsupervised manner. This is particularly advantageous in situations where labelled data is either limited or costly to procure, which is exactly the case of financial corpus. Through node embedding and relation embedding of the graph, Autoencoder can generate useful representations without for explicit supervision.

Autoencoder can perform pre-training on graph datasets and subsequently be fine-tuned for downstream tasks. This approach ensures that the representations encapsulate general graph properties, which can then be transferred to other correlated tasks. Such a transfer learning strategy can bolster the efficacy and performance of graph embedding models.

As shown in Figure 4.7, an Autoencoder consists of two main parts: Encoder and Decoder. The encoder takes the input data and compresses it into a compact, lower-dimensional representation called the "code". It does this by passing the input through one or more hidden layers.

The decoder takes the compressed code from the encoder and reconstructs the original input data as closely as possible. It also passes the code through one or more hidden layers.

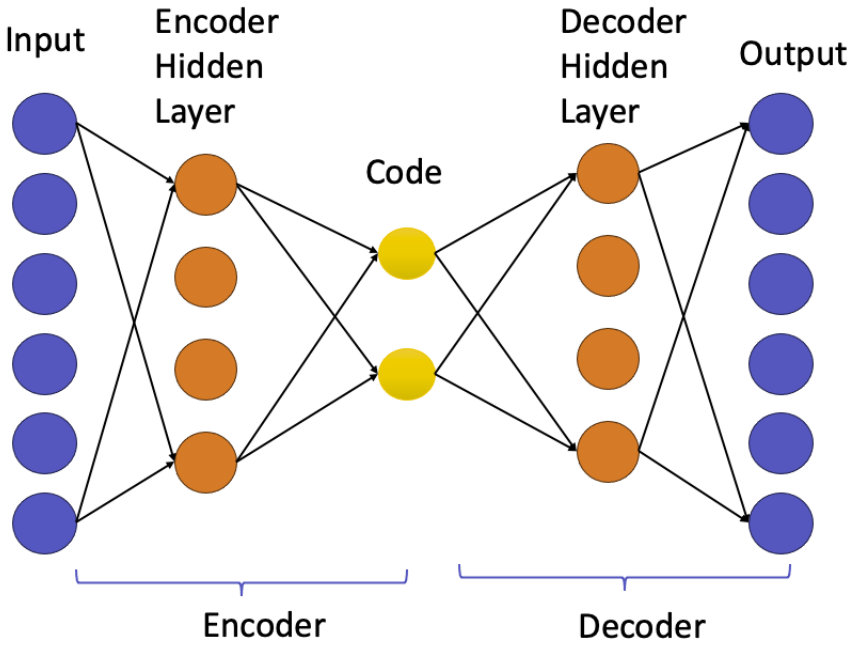

Figure 4.7: Architecture of Autoencoder. Each neuron in the input layer represents an input vector. The encoder is responsible for compressing the input data into a lower-dimensional representation. Code is the central layer in the Autoencoder, which represents the compressed representation of the input data. The decoder reconstructs the input data from the compressed representation provided by the Code. The lines between the neurons represent weights.

Given an input vector x, the encoding and decoding processes can be represented as:

$$h = f(x) = \sigma(W_e x + b_e) \tag{4.18}$$

where W_e is the weight matrix of the encoder.b_e is the bias of the encoder.σ is the activation function.

$$x' = g(h) = \sigma(W_d h + b_d) \tag{4.19}$$

where W_d is the weight matrix of the decoder. b_d is the bias of the decoder.

The main objective of an autoencoder is to minimize the difference between the input x and its reconstruction x'. This is typically done using the Mean Squared Error (MSE) loss for continuous data:

$$\mathcal{L}(x, x') = \frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2 \tag{4.20}$$

where n represents the number of input samples.x_i is the actual input.x'_i is the reconstructed input.

To minimize this loss, the weights W_e and W_d of the encoder and decoder are adjusted using optimization algorithms like gradient descent.

Transforming text data into a graph format offers a structural representation where relationships between sentences can be quantified. This is advantageous for several reasons. Graphs can capture the relational context between sentences that might be missed in a simple vector space. This is usually ignored in other embedding technique while the relations between sentences in financial phrase bank is important since, as stated in Malo et al. (2014), similar sentiments are usually annotated for comments regarding the same finance topic or target companies.

In the graph, nodes represent sentences, and edges can represent semantic or syntactic relationships. Techniques like Graph Neural Networks (GNNs) or Node2Vec (Grover & Leskovec, 2016) can be employed on such graphs to generate embeddings that consider both node attributes and topological structures. These embeddings can potentially capture deeper contextual information.

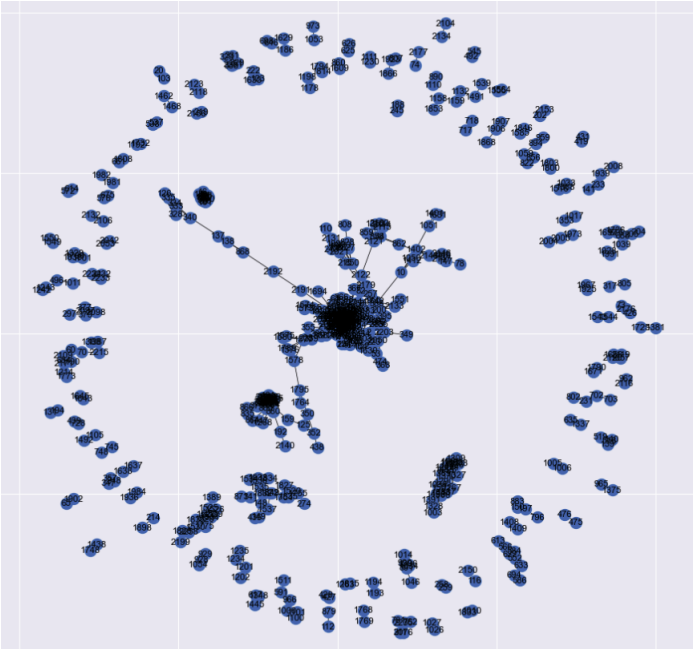

Figure 4.8: Graph representation of Financial Phrase Bank (TF-IDF). Each node represents a sentence from the Financial Phrase Bank. The connections between nodes represent similarities between sentences.

To better understand the conversion process, we take the basic TF-IDF word representation as an example. Figure 4.8 shows the visualization of TF-IDF-based text conversion. To convert our texts to graphs, we first add each sentence as a node and add edges between sentences that have a similar sentiment. The weight of edges is determined by TF-IDF values. Furthermore, we also combine the nodes with sentiment attributes. Take sentence 1 as an example, its node representation includes the following attributes: {sentence: "For the last quarter of 2010 , Componenta 's n...", sentiment: "positive"}. Then, we are able to separate the nodes based on sentiment class and connect nodes with similar semantic features and sentiment. The ring in the figure indicates that a large part of sentences have high similarities, we propose two reasons to explain this phenomenon: firstly, as mentioned in Chapter 3, the dataset reflects a practical scenario and most comments are given neutral sentiment; secondly, some common jargon are widely used in sentences, such as "operating profit". Moreover, we can also see that hardly

any isolated nodes exist in the graph, which indicates that most sentences have some level of similarity with at least one other sentence.

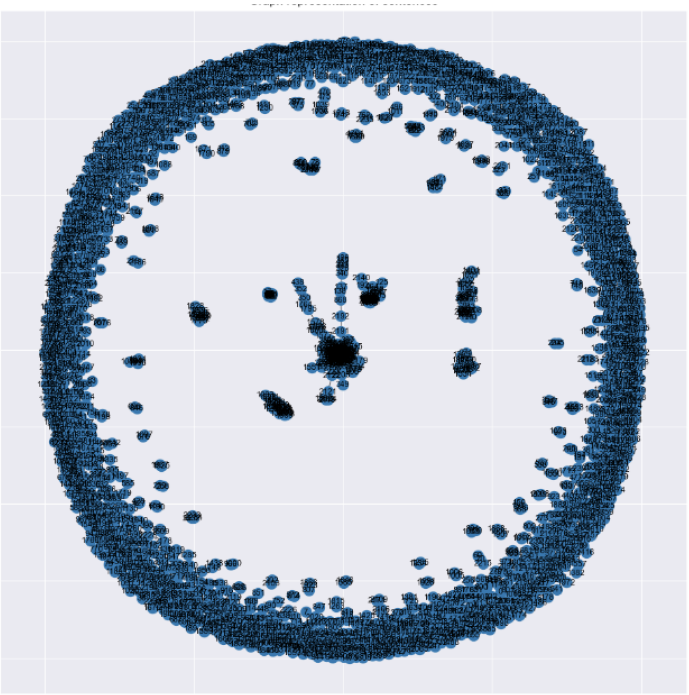

Figure 4.9: Graph representation of Financial Phrase Bank (GloVe)

Similarly, we can conduct the same steps above to other word representations by replacing the calculation methods of the weight of edges with other techniques. Take GloVe as an example, the visualization of its conversion is shown in Figure 4.9. Compared to Figure 4.8, the ring is more dense, which means under pre-trained GloVe models, similarity between sentences is identified as higher.

After the conversion, Encoder part can compress both semantic and sentiment information into a reduced dimension space. And decoder will reconstruct the original features from the compresses representation. Though minimize the loss function, Autoencoder can learn the optimal way to represent the corpora. We will try to use the above method to contrast Autoencoder-enhanced models and improve performance of GloVe, FastText and GloVe.

4.6 Evaluation Metrics

In order to conduct a comprehensive comparative analysis of the embedding techniques we mentioned above, we will use two classes of evaluation metrics for assessment, which are known as intrinsic and extrinsic evaluation.

4.6.1 Intrinsic Evaluation

Intrinsic evaluations aim to assess the quality and coherence of semantic vector space generated by certain embedding techniques. In our research, we use word similarities of manually selected words as our intrinsic evaluation metric.

Word Similarity

Word similarity is a measurement of the semantic closeness between two words. In the context of financial sentiment analysis, understanding word similarity is crucial, as the financial domain is replete with nuanced terminology where slight differences in word choice can lead to significant shifts in meaning.

Word similarity is not just about syntactic or morphological similarity but delves deeper into the semantic realm. For instance, while "stock" and "equity" might not look or sound similar, they are semantically close in the financial context. Moreover, The similarity between two words can be context-dependent. For example, "interest" in a financial context might be closer to "rate" than to "hobby".

This metric quantifies the cosine of the angle between two non-zero vectors in an inner product space. When applied to embeddings, cosine similarity offers a measure of how semantically similar two words are based on their vector representations. Specifically, a cosine similarity value of 1 suggests that the vectors are identical in orientation, indicating high similarity, while a value of 0 implies that the vectors are orthogonal, denoting dissimilarity. In vector space models, where words are represented as vectors, the similarity between two words w_1 and $w_2$can be computed using the cosine similarity:

$$\text{similarity}(w_1, w_2) = \frac{w_1 \cdot w_2}{||w_1||_2 \times ||w_2||_2} \tag{4.21}$$

where $w_1 \cdot w_2$ is the dot product of the two vectorized words, and $||w_1||_2$ and $||w_2||_2$ are their respective L2 norms.

To compute the cosine similarity between word pairs using various embeddings, a consistent methodology is employed. For each word in a given pair, its embedding was extracted using a pre-defined function tailored to the specific embedding technique. These embeddings are then refined, when applicable, using Autoencoder to enhance their representational power.

To compare the word similarity, we manually select a series of word pairs with close or exactly the same meanings in most sentences in the Financial Phrase Bank. The selected word pairs are shown in Table 4.1. Word similarity value 1 indicates that the embeddings of the two words are identical, and a 0 value indicates the two represented words are completely different.

4.6.2 Extrinsic Evaluation

Extrinsic evaluation metrics aim at evaluating the performance of word vectors generated by different embedding techniques in our machine learning model for sentiment analysis. In the domain of sentiment analysis, the performance of models is often evaluated using a suite of metrics that provide a comprehensive understanding of their strengths and weaknesses, namely, accuracy, precision, recall, and F1-score.

Accuracy

Accuracy provides a general measure of how often the model's predictions are correct. Mathematically, it is the ratio of correctly predicted instances to the total instances. The calculation

Word 1	Word 2
profit	earnings
sales	revenue
growth	increase
market	industry
investment	capital
shares	stock
operating	business
financial	economic
quarter	year
euro	EUR
million	mn
global	worldwide
loss	deficit

Table 4.1: Word Pairs. All the word pairs have the same or similar meanings in our context and are selected manually from Financial Phrase Bank sentences. "euro" and "EUR", "million" and "mn", "global" and "worldwide" are pairs of words with exactly the same meanings in all sentences in our dataset.

is as below:

$$\text{Accuracy} = \frac{\text{Number of correctly classified comments}}{\text{Total number of comments}} \tag{4.22}$$

Precision, recall and F1-score

The precision of the model is calculated by dividing the total number of positive predictions by the number of correct positive predictions. It basically sums up the model's capacity to refrain from classifying a negative sample as positive. Recall indicates the model's capacity to detect all positive samples by counting the number of accurate positive predictions it made relative to all real positives. The harmonic mean of recall and precision, or F1-Score, provide a balanced score between the two metrics above. When there is an imbalance in the distribution of classes, it is very helpful. In our instance, as shown in Figure 3.1, the majority of comments are categorized as neutral, which causes the dataset to be unbalanced. The three metrics' mathematical expressions are displayed below.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.23}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.24}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.25}$$

The combination of these metrics provides a comprehensive assessment of the model's extrinsic performance. While accuracy gives a general idea, precision, recall, and F1-score provide insights into the financial sentiment classification model's performance in each sentiment class. To better understand the overall performance in this multi-classification problem, we also introduce macro average and weighted average as extrinsic evaluation metrics.

Weight-Averaging and Macro-Averaging

Besides the extrinsic evaluations account for each sentiment class, we also introduce Macro Average and Weight Average metrics to obtain a holistic perspective of the results. These two metrics are commonly used in evaluating classification machine learning tasks, especially in the evaluation of embeddings' performance in sentiment analysis (Liu, 2017).

Macro Average calculates the extrinsic evaluation metrics for each class separately before averaging them., treating all classes equally. For instance, macro-average precision would be the average of precision values for 'negative', 'neutral', and 'positive' sentiments.

$$\text{Macro-Average Metric} = \frac{1}{n}\sum_{i=1}^{n} m_i \tag{4.26}$$

where n is the number of classes and, in our case, equal to 3; m stands for the extrinsic evaluation metrics, which are recall, precision, accuracy and F1-score.

The weighted average is determined by calculating the metrics for each label and dividing the average by the total number of true instances for each label. It can be more informative than the macro average when dealing with class imbalances in our dataset.

$$\text{Weighted-Average Metric} = \frac{\sum_{i=1}^{n} w_i \cdot m_i}{\sum_{i=1}^{n} w_i} \tag{4.27}$$

where n is the number of classes and also equal to 3; w_i is the number of true instances in class i; m stands for the extrinsic evaluation.

In the context of evaluating embedding performance in sentiment analysis, these metrics are pivotal. Different embeddings might capture various linguistic nuances, and these metrics help in understanding which embeddings are more adept at distinguishing between sentiments. By comparing the values, one can discern which embeddings offer a more balanced performance across sentiments and which ones might be more skewed towards a particular sentiment.

4.7 Multiclass Classification Algorithm

The downstream NLP task that our research revolves around is financial sentiment analysis, which is fundamentally a multiclass classification task. To construct our model, we plan to utilize two traditional classification algorithms: Support Vector Machine and Logistic Regression. This section is dedicated to providing an overview of both models.

4.7.1 Support Vector Machine

For the purpose of training the classification model in our sentiment analysis, Support Vector Machine (SVM) is applied. SVM is a supervised machine learning algorithm that can be used for multiclass classification. SVM works especially well for classifying complicated but small-sized datasets like the Financial Phrase Bank in our research. The main idea under SVM is to find the best hyperplanes to separate data points. The points that are closest to the hyperplane in terms of statistical distance are called support vectors. In basic form, as shown in Figure 4.10, by finding the maximum margin between the positive margin hyperplane and negative margin

hyperplane, the optimal hyperplane in the middle for classification can be determined. For such a linear SVM with a hyperplane defined by is given:

$$d(x) = \frac{|w \cdot x + b|}{\|w\|} \tag{4.28}$$

where w is weight vector and b is bias term, $\cdot$ denotes the dot product and $\|w\|$ is the Euclidean norm of the vector w, d is statistical distance of a point x to the hyperplane.

In our case, where we have three sentiments, binary classification is not efficient. We take advantage of multiclass classification with SVM.

It is possible to extend basic SVM to handle multiclassification problems. In our research, we use the One-to-One approach for sentiment identification. In the One-to-One classification strategy, a distinct hyperplane is constructed to delineate each pair of classes while disregarding data points from any third class. This implies that the separation is exclusively influenced by the data points of the two classes under consideration. To illustrate, as shown in Figure 4.11, the hyperplane distinguishing the green and blue classes is optimized based solely on the distribution of the green and blue data points, without any consideration for the orange data points. In our case, SVM will necessitate three classifiers similar to classifiers in binary classification situations.

In order to measure misclassification and maximize margin, the hinge loss is used. The hinge loss for a single instance is:

$$\mathcal{L}_i = \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \tag{4.29}$$

where $\mathcal{L}_i$ is the loss for the i^{th} instance.$f(x_i; W)_j$ is the score for the j^{th} class of instance x_i given weight matrix W.y_i is the true class of the i^{th} instance. Δ is the margin, a hyperparameter that determines how much the correct class score should exceed the other class scores.

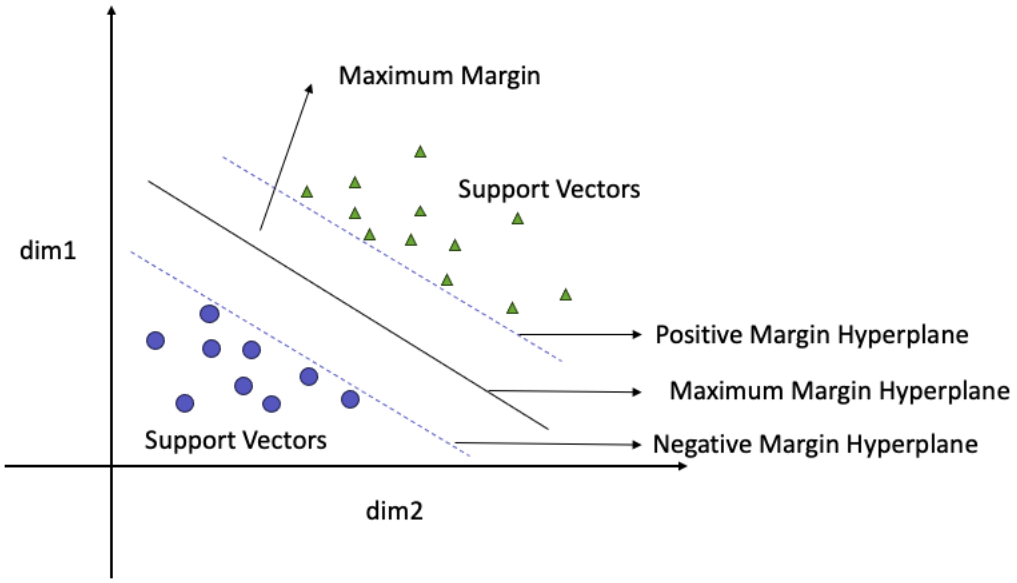

Figure 4.10: Support Vector Machine (SVM) Algorithm

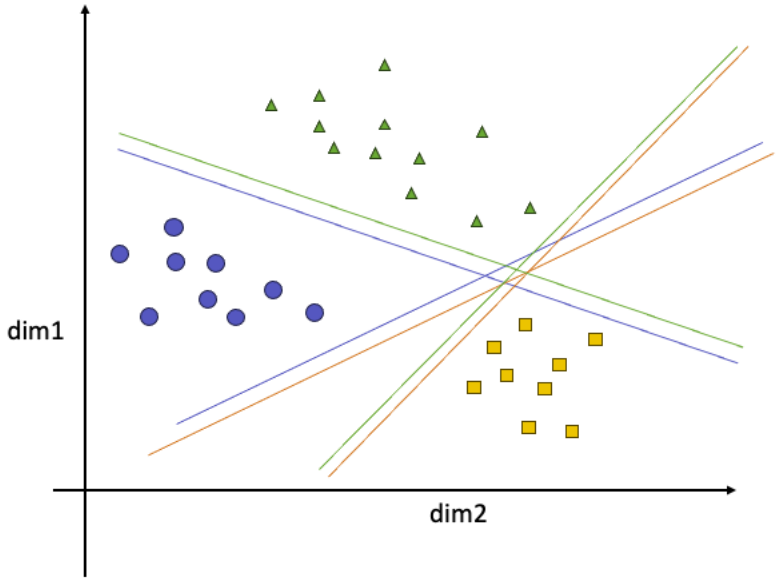

Figure 4.11: Multiclass Classification with SVM using One-to-One approach

In the stage of tuning, we applied grid search with 10-fold Cross-Validation to all of the models and choose the combination of parameters with optimal performance. We split the SVM models into 10 folds, trained the model on 9 folds and validate on the remaining test fold. The following results are based on the performance of the optimal parameter combinations for all models.

SVM has several attributes that make it a relatively more effective and interpretable method for our financial sentiment analysis. Firstly, the embeddings all result in high-dimensional semantic spaces and SVMs are proven to be effective in handling high-dimensional data. Secondly, based on the basics of SVMs that only focus on measuring statistical distance, SVM can be adept at a small-sized dataset, which is helpful in the finance realm with sparsity of data. Moreover, the availability of different kernels in SVMs makes it possible to handle non-linear relationships.

4.7.2 Logistic Regression

Logistic Regression is a popular and basic machine learning tool mainly for classification. Its basic form, binary logistic classification, assumes that the log odds of the target variable is a linear combination of the independent variables. The odds of an event is the ratio of the probability of the event happening to the event not happening.

$$\text{odds} = \frac{p}{1-p} \tag{4.30}$$

The log-odds we mentioned before is the logarithm of the odds.

$$\text{log-odds} = \log\left(\frac{p}{1-p}\right) \tag{4.31}$$

Based on the assumption, the logistic regression function maps the input vectors to a value between 0 and 1, which can be regarded as the probability in the above equations. Mathematically, the logistic function is as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.32}$$

where z is the linear combination of predictors, namely, $z = w^T x$ and w is a parameter vector. w is determined through Maximum Likelihood Estimation, namely, the training process aims at finding the parameters that have the highest probability of observing the given target set of classes.

In our study, we will use multiclass logistic regression to train the models to classify vectorized sentences into three sentiment classes. Based on the same assumptions, multiclass logistic regression used the softmax function to map input. Similar to the function above, the softmax function takes in the vector of logits and converts them to probabilities. The softmax function is:

$$\hat{p}_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4.33}$$

where z_j is the score for class j and K is the number of classes.

In both logistic regressions, the cross-entropy is used as the loss function to measure the difference between the predicted probabilities and the actual class. The loss function is:

$$L(y, \hat{p}) = -\sum_{j=1}^{K} y_j \log(\hat{p}_j) \tag{4.34}$$

where y_j is the true label and $\hat{p}_j$ is the predicted probability for class j.

Logistic regression has several advantages in our study. Firstly, the linearity of logistic regression makes it relatively simple and interpretable. It is beneficial to use a straightforward linear model so that any differences in performance can be attributed more directly to the embeddings themselves rather than the complexities of the model. Moreover, using logistic regression, which has fewer hyperparameters and architectural decisions, makes the tuning process easier and more computationally efficient.

Chapter 5

Results

5.1 Extrinsic Evaluation Results

Table 5.1: Comparison of Embedding Techniques by Accuracy in SVM and Logistic Models

Embedding Technique	**SVM model Accuracy**	**Logistic model Accuracy**
ELMo	0.935982346	0.942604857
GPT-2	0.927152318	0.927152318
BERT	0.918322296	0.916114790
autoBERT	0.905077263	0.905077263
GloVe	0.783664462	0.788079470
FastText	0.759381898	0.761589404
Sense2Vec	0.746136872	0.758995735
Skip-Gram	0.739514349	0.743929360
autoGloVe	0.732891832	0.732891832
autoFastText	0.715231788	0.715231788
CBOW	0.653421634	0.666666667

As shown in Table 5.1, in the comparative analysis of various embedding techniques, biLSTM-based ELMo emerges as the top-performing model in terms of accuracy, which is approximately 93.60% in the SVM model and 94.26% in the Logistic model. This is closely followed by the transformer-based models GPT-2 and BERT, which have accuracies of 92.72% and 91.83% in the SVM model and 92.72% and 91.61% in the Logistic model, respectively. This means contextualized embedding techniques significantly outperform other embedding techniques in capturing the nuances in financial English and classifying sentiment in the finance domain.

The ranking of overall accuracy is in line with the complexity of the architecture of the models and their popularity in the industry. Word2Vec (CBOW and Skip-Gram) are proven to have the lowest overall accuracy, which are 65.34% and 73.95% for the SVM model and 66.67% and 74.39% for the Logistic model, respectively. Sense2Vec, the model inspired by Word2Vec, slightly improve the accuracy to 74.61% and 75.90%. Similarly, FastText also shows improvement due to taking OOV into account and raising the accuracy to 75.94% and 76.15%. GloVe, owing to the abundant external resources, the accuracy outperforms Word2Vec and reaches 78.36% and 78.81%. It is also worth noticing that autoencoder-enhanced models do not show our expected improvement in extrinsic performance, and all have relatively lower accuracy compared to the

Metrics	Keypoints
Negative Precision	CBOW and Skip-Gram fail to classify negative sentiment. Hence, they have no values in all precision, recall and f1-score of negative sentiment. BERT model boasts an impressive negative precision and outperforms GPT-2 and ELMo.
Neutral Precision	BERT consistently outperform both ELMo and GPT-2.
Positive Precision	GloVe and FastText display heightened precision for positive sentiments. Among contextualized embedding, BERT and ELMo are close in accurately identifying positive sentiments.
Negative Recall	ELMo emerges as the leader in this metric, reflecting its adeptness at correctly capturing negative sentiments.
Neutral Recall	Among the contextualized embeddings, ELMo and BERT exhibit the strongest performances, surpassing GPT-2.
Positive Recall	ELMo and BERT show robust performances, with values that are higher than word, sense, and autoencoder-enhanced embedding techniques
Negative F1-score	ELMo's dominance is evident, highlighting its balanced precision and recall for negative sentiments.
Neutral F1-score	Performance of BERT and ELMo suggest a balance between precision and recall for neutral sentiments. The conventional models, especially FastText, demonstrate commendable scores as well.
Positive F1-score	GloVe emerges as a strong performer among traditional models. ELMo and BERT still have a good performance
Macro and Weighted Averages	When it comes to macro-averaged precision, recall, and F1-scores, the contextualized embeddings, especially ELMo and BERT, clearly outperform. The weighted averages further underline the consistency of models like ELMo and BERT across different sentiment classes.

Table 5.2: Key findings from Precision, Recall and F1-score

original models. Another notable observation in this ranking is that the ranking is not affected by different modelling algorithms, both SVM and Logistic Regression models show exactly the same accuracy ranking for different embedding techniques. We propose that the extrinsic performance of embedding techniques is consistent and robust in the downstream financial sentiment analysis in terms of overall accuracy.

Table C.1 to C.6 in Appendix C shows the detailed results of all extrinsic evaluation metrics. For precision, recall and F1-score, we summarize the key findings in Table 5.2.

5.2 Intrinsic Evaluation Results

Table 5.3, 5.4, 5.5 show the result of word similarities.

A key observation is that even though original models outperform Autoencoder-enhanced models in terms of extrinsic evaluation metrics, we observe a salient improvement in word similarities in refined FastText and BERT. The word pairs "euro" and "EUR", "million" and "mn", "global" and "worldwide" are supposed to have exactly the same semantics in all sentences and the word similarities of the three word pairs theoretically should equal 1. The autoencoder-enhanced gives values significantly closer to 1 compared to the original versions. It means

the Autoencoder does have the capability of capturing the nuances between certain words and measuring the word similarity in a better manner than the original models. However, we also notice that Autoencoder-enhanced FastText has a significantly lower score for the word pair "sales" & "revenue" compared to the original FastText. This hints that Autoencoder-enhanced embedding techniques might not always provide a comprehensive improvement for all dimensions of word similarity.

Other than the first finding, we also notice that traditional word embedding techniques, Word2Vec fails to measure similarities between word pair "loss" and "deficit" due to the low frequencies of the word "deficit" in our corpora. GloVe fails to measure similarities between "EUR" and "euro" and it is likely because of OOV problem in pre-trained word vectors.

Moreover, similar to extrinsic evaluation comparison, contextualized embedding techniques still have overall higher performance than the other embedding techniques. Embeddings extracted from GPT-2 excel across almost all word pairs, with scores typically above 0.9. This indicates its prowess in understanding word semantics and capturing synonymous relationships. However, while ELMo exhibits strong results in extrinsic evaluations, it lags behind in capturing synonymous relationships compared to BERT and GPT-2. This observation is particularly evident from ELMo's word similarity scores for certain synonyms, where it produces values considerably less than the theoretical ideal of 1. We propose this is because the transformer's attention mechanism in BERT and GPT-2 might be better suited to understanding nuanced relationships between words, especially in contexts where words are used interchangeably. It may also be because some abbreviations like "mn" and "euro" do not commonly occur in the corpora that ELMo is pre-trained on.

Table 5.3: Word Similarity of Word Embeddings

Word Pairs		Skip-Gram	CBOW	GloVe	FastText
profit	earnings	0.80	0.99	0.89	0.57
sales	revenue	0.89	0.99	0.78	0.64
growth	increase	0.98	0.99	0.77	0.44
market	industry	0.95	0.99	0.72	0.62
investment	capital	0.97	0.99	0.65	0.60
shares	stock	0.95	0.99	0.85	0.61
operating	business	0.82	0.99	0.61	0.51
financial	economic	0.94	0.97	0.77	0.74
quarter	year	0.94	0.99	0.69	0.57
euro	EUR	0.76	0.97		0.63
million	mn	0.89	0.99	0.23	0.48
global	worldwide	0.99	0.99	0.72	0.67
loss	deficit			0.60	0.46

Table 5.4: Word Similarity of Contextualized Embeddings

Word Pairs		BERT	ELMo	GPT-2
profit	earnings	0.85	0.76	0.98
sales	revenue	0.81	0.71	0.99
growth	increase	0.85	0.54	0.94
market	industry	0.80	0.64	0.96
investment	capital	0.79	0.41	0.98
shares	stock	0.61	0.71	0.98
operating	business	0.84	0.43	0.99
financial	economic	0.91	0.69	0.99
quarter	year	0.66	0.60	0.99
euro	EUR	0.75	0.58	0.99
million	mn	0.76	0.48	0.97
global	worldwide	0.89	0.75	0.99
loss	deficit	0.77	0.56	0.98

Table 5.5: Word Similarity of Autoencoder-enhanced Embeddings

Word Pairs		autoFastText	autoGloVe	autoBERT
profit	earnings	0.68	0.86	0.94
sales	revenue	0.67	0.66	0.92
growth	increase	0.52	0.75	0.93
market	industry	0.71	0.83	0.90
investment	capital	0.63	0.74	0.90
shares	stock	0.66	0.79	0.92
operating	business	0.53	0.39	0.93
financial	economic	0.72	0.82	0.95
quarter	year	0.73	0.76	0.86
euro	EUR	0.71		0.85
million	mn	0.62	0.30	0.88
global	worldwide	0.77	0.76	0.94
loss	deficit	0.61	0.65	0.89

Chapter 6

Discussion and Conclusion

6.1 Discussion and Further Explanation of the Results

Generally speaking, contextualized embedding is believed to have better performance in word representation. Among the three contextualized embedding techniques, the performance of ELMo is particularly impressive compared to Transformer-based models. A similar phenomenon also occurs in Schlechtweg, McGillivray, Hengchen, Dubossarsky and Tahmasebi (2020), where the authors also detect ELMo's outperformance over transformer-based models. We propose that such outperformance may come from several perspectives: Firstly, ELMo typically adds its pre-trained representations to the task-specific model and only fine-tunes the task-specific parameters, while BERT and GPT-2 fine-tunes the entire model. This means that ELMo is less prone to overfitting when adapting to the specific sentiment analysis task, especially when the amount of labelled data is limited. Secondly, implementation details or other architectural nuances can make a difference in performance. Some configurations might favour ELMo architecture over BERT.

However, while ELMo architecture is proven to have a better extrinsic performance in the SVM and logistic regression sentiment analysis model, it is computationally more intensive and slower in processing compared to its transformer counterparts. On the other hand, BERT and GPT-2, despite their slightly lower accuracy metrics, BERT still outperform in a series of other metrics and also offer faster processing speeds. Moreover, embedding from GPT-2 also dominantly outperforms all the other embedding techniques in terms of intrinsic evaluation. More importantly, transformer-based models are designed to handle a broader range of linguistic structures due to their advanced architectures. In conclusion, The flexibility of transformer models and better performance in intrinsic evaluation make transformer-based contextualized embedding techniques more adaptable to various tasks and datasets in the rapidly changing finance realm.

Several factors can be attributed to extrinsic evaluation ranking. The superior performance of models like ELMo, GPT-2, and BERT can be attributed to their intricate architectures that leverage deep transformer mechanisms and LSTM and are adept at capturing contextual information. Additionally, the vast datasets on which these models are pre-trained enable them to discern a wide array of linguistic patterns. In contrast, models like GloVe or FastText generate static embeddings for each word, which might limit their efficiency.

A salient observation from the analysis is the relatively subdued performance of Autoencoder-enhanced models compared to their original versions in terms of extrinsic evaluation but a relatively better performance in terms of intrinsic evaluation. The performance of Autoencoder-enhanced BERT, the highest among the Autoencodee-enhanced models, shows a significant improvement in word similarities while lagging behind its original version in accuracy. This trend is consistent in FastText as well. We propose that such intrinsic improvement comes from Autoencoder-based models' deeper understanding between sentences through modelling node relations. However, information loss during the conversion from text to graph causes a decrease in extrinsic performance. Autoencoders operate by compressing information into a condensed representation and subsequently reconstructing the original data. This compression and reconstruction process, while capturing non-linearities and intricate patterns, might also lead to potential information loss. The added layer of complexity introduced by Autoencoders does not always guarantee enhanced performance in downstream tasks, especially if the original embeddings were already task-optimized. Furthermore, the training intricacies of Autoencoders, which demand meticulous tuning, can sometimes result in the model overlooking essential data features. It is also worth noting that autoencoders might not be inherently optimized for specific NLP tasks in the same vein as models explicitly designed for language processing, such as BERT or ELMo. However, there is no reason that we should ignore the potential in Autoencoder-based embedding models and graph embedding's applications in text data due to a slight decline in extrinsic performance.

Our answer to the central question of this research is: Generally, contextualized embedding techniques are proven to have a better performance in financial sentiment analysis models in terms of both extrinsic and intrinsic evaluation. Within contextualized embedding, ELMo has slightly better extrinsic performance, such as accuracy, while showing significantly poorer performance in intrinsic evaluation. BERT and Embeddings from GPT-2 show high performance in both extrinsic and intrinsic evaluations. Sense embedding can improve extrinsic performance in a limited range but it lacks an intrinsic evaluation method. Moreover, it is possible to apply graph embedding and autoencoder for word representation, even though intrinsic performance is proven to be better, but the performance in sentiment analysis can not be guaranteed.

6.2 Limitations and Potential Future Research

There are several significant limitations in our research and they can pave the way for further research.

Even though financial English is widely used across the global finance industry, we still cannot ignore that non-English-speaking economies are playing an increasingly important role in the world's financial system, such as the European Union. In our research, only English comments are contained. Comparative analysis of embedding techniques in financial sentiment analysis regarding multilingual corpus still remains void, and there is a research gap in comparing multilingual embedding techniques, such as M-BERT. Besides the linguistic perspective, more embedding techniques can be added to the comparison, we only selected several representative techniques in our research.

Another deficiency in this research is that the correlation between extrinsic and intrinsic

evaluation metrics still needs to be explored. In our research, while intrinsic evaluation focuses on the semantic relation between limited selected words, extrinsic evaluation focuses more on the overall performance of the models. An embedding technique that perfectly captures word similarities may not provide sufficient information for sentiment classification. Furthermore, embedding is just one part of the entire modelling process. Factors in other stages, such as tuning, may also affect our extrinsic evaluation.

Moreover, pre-trained models' performance highly depends on the external word vector resources. Lack of universal training corpora may lead us to insufficient comparison results. We suggest that, if possible, universal training datasets should be constructed and build up finance domain-specific word vectors. Such pre-trained models will exclusively boost the credibility of the comparative analysis.

References

Adhikari, S., Thapa, S., Naseem, U., Lu, H. Y., Bharathy, G. & Prasad, M. (2023). Explainable hybrid word representations for sentiment analysis of financial news. *Neural Networks*, *164*, 115–123.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*.

Baker, M. & Wurgler, J. (2006). Investor sentiment and the cross-section of stock returns. *The journal of Finance*, *61*(4), 1645–1680.

Bausch, P. & Bumgardner, J. (2006). Make a flickr-style tag cloud. *Flickr hacks*, 82–86.

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, *5*, 135–146.

Dang, N. C., Moreno-García, M. N. & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, *9*(3), 483.

Davis, A. K., Piger, J. M. & Sedor, L. M. (2012). Beyond the numbers: Measuring the information content of earnings press release language. *Contemporary Accounting Research*, *29*(3), 845–868.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dudley-Evans, T. & St John, M. J. (1998). *Developments in english for specific purposes: A multi-disciplinary approach*. Cambridge university press.

Firth, J. R. (1957). Ethnographic analysis and language with reference to malinowski's views. *Man and Culture: an evaluation of the work of Bronislaw Malinowski*, 93–118.

Frieder, L. & Zittrain, J. (2007). Spam works: Evidence from stock touts and corresponding market activity. *Hastings Comm. & Ent. LJ*, *30*, 479.

Grover, A. & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 855–864).

Kaneko, M. & Bollegala, D. (2020). Autoencoding improves pre-trained word embeddings. *arXiv preprint arXiv:2010.13094*.

Kashyap, A. K. & Stein, J. C. (2023). Monetary policy when the central bank shapes financial-market sentiment. *Journal of Economic Perspectives*, *37*(1), 53–75.

Li, J. & Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070*.

Liu, H. (2017). Sentiment analysis of citations using word2vec. *arXiv preprint arXiv:1704.00177*.

Loughran, T. & McDonald, B. (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of finance*, *66*(1), 35–65.

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

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Malandri, L., Xing, F. Z., Orsenigo, C., Vercellis, C. & Cambria, E. (2018). Public mood–driven asset allocation: The importance of financial sentiment in portfolio management. *Cognitive Computation*, *10*, 1167–1176.

Malo, P., Sinha, A., Korhonen, P., Wallenius, J. & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, *65*(4), 782–796.

McEnery, A. & Baker, P. (2015). *Corpora and discourse studies: Integrating discourse and corpora*. Springer.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Yih, W.-t. & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 746–751).

Nickerson, C. (2005). *English as a lingua franca in international business contexts* (Vol. 24) (No. 4). Elsevier.

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). *Deep contextualized word representations.*

Pilehvar, M. T. & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning*. Morgan & Claypool Publishers.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Rawte, V., Gupta, A. & Zaki, M. J. (2020). A comparative analysis of temporal long text similarity: Application to financial documents. In *Workshop on mining data for financial applications* (pp. 77–91).

Salton, G., Wong, A. & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Schlechtweg, D., McGillivray, B., Hengchen, S., Dubossarsky, H. & Tahmasebi, N. (2020). Semeval-2020 task 1: Unsupervised lexical semantic change detection. *arXiv preprint arXiv:2007.11464*.

Schnabel, T., Labutov, I., Mimno, D. & Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in*

natural language processing (pp. 298–307).

Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics*, *24*(1), 97–123.

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

Tetlock, P. C., Saar-Tsechansky, M. & Macskassy, S. (2008). More than words: Quantifying language to measure firms' fundamentals. *The journal of finance*, *63*(3), 1437–1467.

Trask, A., Michalak, P. & Liu, J. (2015). sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Appendix A

Word Cloud

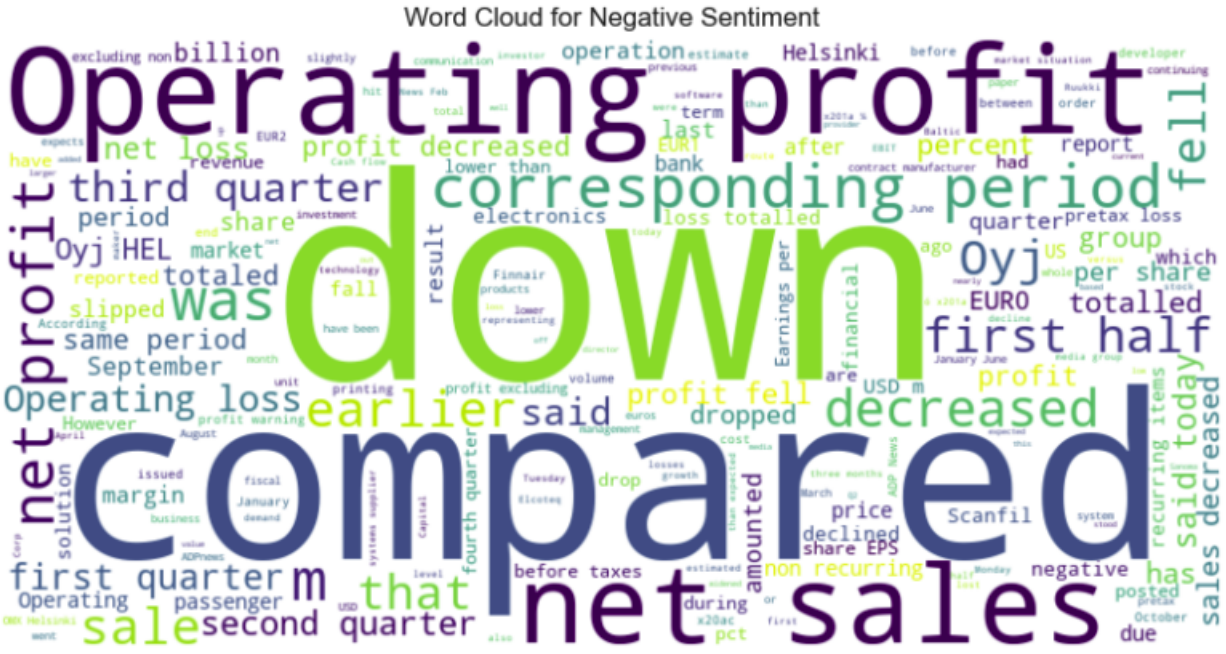

Figure A.1: Word Cloud for Comments with Negative Comments

Figure A.2: Word Cloud for Comments with Positive Comments

Figure A.3: Word Cloud for Comments with Neural Comments

Appendix B

Implementation of Embedding Techniques

Table B.1: Implementation of Embedding Techniques

Embedding Technique	**Pre-trained model**	**Description**
Skip-Gram	NaN	Custom-trained embeddings based on Financial Phrase Bank.
CBOW	NaN	Custom-trained embeddings based on Financial Phrase Bank.
GloVe	glove.6B.300d	Trained on 6 billion tokens. Each vector has a dimensionality of 300.
FastText	wiki-news-300d-1M	Autoencoder 1 million words and phrases in Wikipedia. Each vector has a dimensionality of 300.
autoGlove	glove.6B.300d	Original model is trained on 6 billion tokens. Each vector has Autoencoderality of 300. The model is enhanced with Autoencoder.
autoFastText	wiki-news-300d-1M	Original model is trained on 1 million words and phrases in Wikipedia. Each vector has a dimensionality of 300. The model is enhanced with Autoencoder.

Table B.2: Implementation of Embedding Techniques

Embedding Technique	**Pre-trained model**	**Description**
ELMo	elmo_2x4096_512_2048 cnn_2xhighway_weights	Available in AllenNLP. The model has two LSTM layers and each has 4096 units. Each embedding has a dimensionality of 2048. Both option and weight files are from this model.
BERT	bert-base-uncased	Trained on the BooksCorpus and English Wikipedia with more than 3300 million words in total. It has 110 million parameters, including 12 transformer layers, 768 hidden units, and 12 attention heads. The model treats uppercase and lowercase letters as the same character.
GPT-2	GPT-2 medium	Note GPT-2 is not an embedding technique in essence, we extract embeddings from the GPT-2 tokenizer and model
autoBERT	bert-base-uncased	Original model is trained on the BooksCorpus and English Wikipedia wAutoencoderan 3300 million words in total. It has 110 million parameters, including 12 transformer layers, 768 hidden units, and 12 attention heads. The model treats uppercase and lowercase letters as the same character. The model is enhanced with Autoencoder.

Appendix C

Extrinsic Evaluation

Table C.1: Word Embedding Extrinsic Evaluation Metrics Comparison for SVM models

Metric	**CBOW**	**Skip-Gram**	**GloVe**	**FastText**
Overall accuracy	0.653421634	0.739514349	0.78366446	0.759381898
negative precision	0	0	0.52631579	1
negative recall	0	0	0.35087719	0.01754386
negative f1-score	0	0	0.42105263	0.034482759
negative support	57	57	57	57
neutral precision	0.6674937965	0.783382789	0.85906040	0.839228296
neutral recall	0.9817518248	0.96350365	0.93430657	0.952554745
neutral f1-score	0.7946824225	0.864157119	0.89510490	0.892307692
neutral support	274	274	274	274
positive precision	0.54	0.612068966	0.67521368	0.581560284
positive recall	0.221311475	0.581967213	0.64754098	0.672131148
positive f1-score	0.313953488	0.596638655	0.66108787	0.623574144
positive support	122	122	122	122
macro avg precision	0.402497932	0.465150585	0.68686329	0.806929527
macro avg recall	0.4010211	0.515156954	0.64424158	0.547409917
macro avg f1-score	0.369545304	0.486931925	0.65908180	0.516788198
macro avg support	453	453	453	453
weighted avg precision	0.549168433	0.638673947	0.76767907	0.790063814
weighted avg recall	0.653421634	0.739514349	0.78366446	0.759381898
weighted avg f1-score	0.565221433	0.683375202	0.77243148	0.711995299
weighted avg support	453	453	453	453

Table C.2: Word Embedding Extrinsic Evaluation Metrics Comparison for Logistic Models

Metric	**CBOW**	**Skip-Gram**	**GloVe**	**FastText**
Overall accuracy	0.666666667	0.743929360	0.788079470	0.761589404
negative precision	0.0	0.5	0.547619048	0.666666667
negative recall	0.0	0.017543860	0.403508772	0.105263158
negative f1-score	0.0	0.033898305	0.464646465	0.181818182
negative support	57	57	57	57
neutral precision	0.679389313	0.788059701	0.864406780	0.811145511
neutral recall	0.974452555	0.963503650	0.930656934	0.956204380
neutral f1-score	0.800599700	0.866995074	0.896309315	0.877721943
neutral support	274	274	274	274
positive precision	0.583333333	0.620689655	0.681034483	0.636363636
positive recall	0.286885246	0.590163934	0.647540984	0.631147541
positive f1-score	0.384615385	0.605042017	0.663865546	0.633744856
positive support	122	122	122	122
macro avg precision	0.420907549	0.636249786	0.697686770	0.704725271
macro avg recall	0.420445934	0.523737148	0.660568897	0.564205026
macro avg f1-score	0.395071695	0.501978465	0.674940442	0.564428327
macro avg support	453	453	453	453
weighted avg precision	0.568033860	0.706738402	0.775161038	0.745894555
weighted avg recall	0.666666667	0.743929360	0.788079470	0.761589404
weighted avg f1-score	0.587830894	0.691620264	0.779393372	0.724451040
weighted avg support	453	453	453	453

Table C.3: Contextualized Embedding Extrinsic Evaluation Metrics Comparison for SVM models

Metric	**BERT**	**GPT-2**	**ELMo**
Overall accuracy	0.918322296	0.927152318	0.93598234
negative precision	0.867924528	0.814814815	0.88
negative recall	0.807017544	0.771929825	0.771929825
negative f1-score	0.836363636	0.792792793	0.822429907
negative support	57	57	57
neutral precision	0.939716312	0.974729242	0.981549815
neutral recall	0.967153285	0.98540146	0.97080292
neutral f1-score	0.95323741	0.980036298	0.976146789
neutral support	274	274	274
positive precision	0.889830508	0.868852459	0.863636364
positive recall	0.860655738	0.868852459	0.93442623
positive f1-score	0.875	0.868852459	0.897637795
positive support	122	122	122
macro avg precision	0.899157116	0.886132172	0.908395393
macro avg recall	0.878275522	0.875394581	0.892386325
macro avg f1-score	0.888200349	0.880560516	0.898738164
macro avg support	453	453	453
weighted avg precision	0.91724788	0.926093282	0.937016083
weighted avg recall	0.918322296	0.927152318	0.93598234
weighted avg f1-score	0.917460878	0.926532306	0.935661227
weighted avg support	453	453	453

Table C.4: Contextualized Embedding Extrinsic Evaluation Metrics Comparison for Logistic Models

Metric	**BERT**	**GPT2**	**ELMo**
Overall accuracy	0.916114790	0.927152318	0.942604857
negative precision	0.905660377	0.830188679	0.916666667
negative recall	0.842105263	0.771929825	0.771929825
negative f1-score	0.872727273	0.8	0.838095238
negative support	57	57	57
neutral precision	0.939716312	0.967509025	0.974637681
neutral recall	0.967153285	0.978102190	0.981751825
neutral f1-score	0.953237410	0.972776770	0.978181818
neutral support	274	274	274
positive precision	0.864406780	0.878048780	0.883720930
positive recall	0.836065574	0.885245902	0.934426230
positive f1-score	0.85	0.881632653	0.908366534
positive support	122	122	122
macro avg precision	0.903261156	0.891915495	0.925008426
macro avg recall	0.881774707	0.878425972	0.896035960
macro avg f1-score	0.891988228	0.884803141	0.908214530
macro avg support	453	453	453
weighted avg precision	0.915149091	0.926137260	0.942858009
weighted avg recall	0.916114790	0.927152318	0.942604857
weighted avg f1-score	0.915303543	0.926490107	0.941752680
weighted avg support	453	453	453

Table C.5: Autoencoded Embedding and Sense Embedding Extrinsic Evaluation Metrics Comparison for SVM models

Metric	**autoFastText**	**autoGloVe**	**autoBERT**	**Sense2Vec**
Overall accuracy	0.715231788	0.732891832	0.905077263	0.74613687
negative precision	0	0	0.897959184	0.40000000
negative recall	0	0	0.771929825	0.07017544
negative f1-score	0	0	0.830188679	0.11940299
negative support	57	57	57	57
neutral precision	0.757225434	0.788343558	0.939501779	0.82467532
neutral recall	0.95620438	0.937956204	0.96350365	0.92700730
neutral f1-score	0.84516129	0.856666667	0.951351351	0.87285223
neutral support	274	274	274	274
positive precision	0.579439252	0.595238095	0.829268293	0.59259259
positive recall	0.508196721	0.614754098	0.836065574	0.65573770
positive f1-score	0.541484716	0.60483871	0.832653061	0.62256809
positive support	122	122	122	122
macro avg precision	0.445554895	0.461193885	0.888909752	0.60575597
macro avg recall	0.4881337	0.517570101	0.857166349	0.55097348
macro avg f1-score	0.462215335	0.487168459	0.871397697	0.53827444
macro avg support	453	453	453	453
weighted avg precision	0.614064807	0.637141683	0.90458696	0.70873584
weighted avg recall	0.715231788	0.732891832	0.905077263	0.74613687
weighted avg f1-score	0.657031631	0.681052956	0.904138407	0.71064192
weighted avg support	453	453	453	453

Table C.6: Autoencoded Embedding and Sense Embedding Extrinsic Evaluation Metrics Comparison for Logistic models

Metric	**autoFastText**	**autoGloVe**	**autoBERT**	**sense2vec**
Overall accuracy	0.6931567329	0.7417218543	0.8984547461	0.7527593819
negative precision	0.5	0.5	0.7666666667	0.5416666667
negative recall	0.0175438596	0.1929824561	0.8070175439	0.2280701754
negative f1-score	0.0338983051	0.2784810127	0.7863247863	0.3209876543
negative support	57	57	57	57
neutral precision	0.7257617729	0.8063492063	0.9460431655	0.803125
neutral recall	0.9562043796	0.9270072993	0.9598540146	0.9379562044
neutral f1-score	0.8251968504	0.8624787776	0.9528985507	0.8653198653
neutral support	274	274	274	274
positive precision	0.5666666667	0.6120689655	0.852173913	0.6513761468
positive recall	0.4180327869	0.5819672131	0.8032786885	0.5819672131
positive f1-score	0.4811320755	0.5966386555	0.8270042194	0.6147186147
positive support	122	122	122	122
macro avg precision	0.5974761465	0.6394727240	0.8549612484	0.6653892712
macro avg recall	0.4639270087	0.5673189895	0.8567167490	0.5826645310
macro avg f1-score	0.4467424103	0.5791994819	0.8554091855	0.6003420448
macro avg support	453	453	453	453
weighted avg precision	0.6545078567	0.7154792414	0.8981921517	0.7293579247
weighted avg recall	0.6931567329	0.7417218543	0.8984547461	0.7527593819
weighted avg f1-score	0.6329674473	0.7174007036	0.8980336214	0.7293368883
weighted avg support	453	453	453	453