Financial Sentiment Analysis

Final Project Report

GRISI Clément
École des Ponts ParisTech
grisi.clement@gmail.com

Abstract

Financial and economic news is continuously monitored by financial market participants and plays an integral part of investment activity [1]. The growing availability of financial texts in the wake of big data has challenged most organizations and led to an ever increasing demand for powerful analytical tools. Among them, sentiment analysis appears as an efficient method for extracting actionable insights from news headlines.

Deep-learning based methods have shattered performance in many challenging natural language processing applications, including sentiment analysis. I take advantage of the present project to familiarize myself with all of these methods, including the recently introduced and popular Transformers [2], investigating their performance for financial sentiment analysis. My code will be publicly available at https://github.com/clementgr/sentiment-analysis

1. Introduction

Stock market movement prediction is a very challenging task due to the large amount of unforseen events that can cause stock prices to change. Today, stock trading involves much more than monitoring stock prices, as traders are generally looking for vast amount of market information, news and reports about the company they wish to buy stocks from. When new information becomes available, all actors in the economy update their positions and prices adjust accordingly.

Financial texts have become more readily available due to the proliferation of postings on the Internet and the ever-increasing demands for market transparency. Dealing with this deluge of data has become increasingly challenging for business analysts. Be it daily news, financial reports, official company statements or tweets, financial texts have emerged as a precious source of information. They appear as a good proxy for the state of a company at a given period of time and carry important information that is of value in any financial decision.

Under the efficient market hypothesis [3], "the efficiency of markets relies on the delivery of market information to the investors in a timely and correct manner". However, as the stream of market data continues to expand rapidly, manual analysis to derive actionable insights largely exceeds human capacities: perfectly informed and rational decisions is no longer attainable. In order to guide the attention of financial analysts over such continuous flow of data, it has become crucial to develop automated text analysis techniques capable of skimming through countless lines of text and picking up on relevant information.

Natural language processing (NLP) techniques are particularly suited to make sense out of such unstructured data. Among these, sentiment analysis has ermerged as an important analytic tool for dealing with the current text boom [1]. By revealing the latest trends in the public mood as reflected in the media, sentiment analysis provides clues for making better decisions in chaotic financial markets. Though NLP has traditionally employed machine learning models for sentiment analysis, the ever increasing amount of textual data available has favored the emergence of deep learning based approaches, which will be the focus of this project.

2. Problem Definition

Sentiment analysis is the task of extracting sentiments or opinions of people from written language [4]. Though it ranges from detecting emotions (e.g., fear, happiness), to sarcasm and intent (e.g., complaints, feedback), I will focus on its simplest form, that is classifying text as positive, negative or neutral.

Given the complexity of human language, sentiment analysis is far from trivial. Financial sentiment analysis is even more challenging due to the use of domain-specific language and the unavailability of large annotated datasets. Let's consider a simple – yet interesting – example to better grasp the possible difficulties of the task:

Shares of food delivery companies surged despite the catastrophic impact of coronavirus on global markets.

The above statement contains a clear positive message about food delivery companies, together with a negative message about global markets. It is difficult for computers to extract this positive message because they would have to crack the financial context and understand what is positive and what is negative from the financial point of view.

Due to the numerous challenges it involves, sentiment analysis in finance has become an important research topic [5, 6, 7, 8, 9]. In spite of being highly unstructured, textual data is certainly not random sequences of characters. Researchers have long treated words as atomic units, simply representing them as indices in a vocabulary. However, such a representation of words comes short in analyzing deeper semantic meaning of a given text: it completely disregards grammar, and doesn't provide any notion of similarity between words. Hence, NLP gradually shifted from discrete word counting to continuous vector representations of words. Deep learning based approaches, which allow for complex feature extraction, location identification, and order information, have greatly contributed to this transition.

3. Related Work

Previous research conducted on sentiment analysis can be divided into two groups: 1) machine learning methods with features extracted from text based on word counting [10, 11, 12], 2) deep learning approaches leveraging distributed representations of words [13, 14, 15].

A survey by Loughran and McDonald [16] recapitulates the recent works on financial text analysis using machine learning with (discrete) bag-of-words approach and lexicon-based methods. For example, the same authors created an expert annotated dictionnary of positive, negative, and neutral financial terms, which enabled them to measure the sentiment of a sentence by counting words with a specific dictionary value. Another important example is [17], which compares various machine-learning binary classifiers to predict tweets sentiments. They show that the SVM classifier is more accurate compared to Decision Trees and Naïve Bayes classifier.

One of the first papers that used deep learning methods for financial sentiment analysis was [18]. Authors apply an long short-term memory (LSTM) neural network to predict stock-market movements from company announcements and show that method to be more accurate than traditional machine learning approaches. Several other works confirm the efficiency of deep learning models for financial sentiment analysis, including recurrent neural network (RNN) [19, 20], convolutional neural networks (CNN) [21, 22] and attention mechanism [23]. Before Transfromers were introduced, the state-of-the-art on the Financial PhraseBank dataset[1] was held by an LSTM [24].

The success of deep learning based methods in NLP relies to a large extent on the introduction and improvement of text representation methods, such as word encoders [25, 26, 27]. These convert words into distributed representations encoding the semantic information contained in words and sentences, which is crucial for sentiment analysis. The current state-of-the-art on the Financial PhraseBank dataset is held by [28], who use a Transformer-based architecture.

4. Methodology

4.1. Dataset

The dataset used is Financial PhraseBank [29], a public dataset for financial sentiment classification. It consists of 4845 english sentences selected randomly from financial news found on LexisNexis database. Annotators with background in finance were asked to label these sentences according to how they think the information contained might affect the mentioned company stock price: positive, negative & neutral. Figure 1 shows the label distribution in the whole dataset.

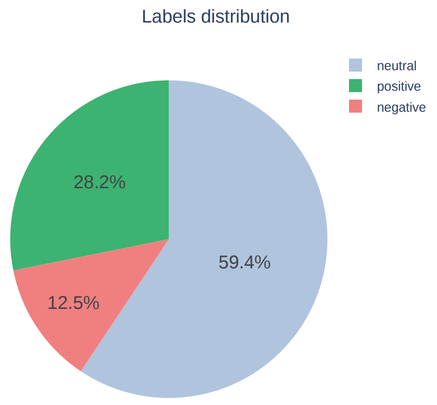


Figure 1: Financial PhraseBank Label Distribution

[1]this is the dataset I have used for this project

In order to be able to compare to results reported on this dataset in the litterature, I used the same train/val/test split as performed by the authors of [28]: they set aside 20% of all sentences as test (970) and 20% of the remaining as validation set (388). In the end, the training set includes 3488 examples. Figure 2 shows the label distribution for each subset: we see that they all follow the distribution of the full dataset, which is a desirable property.

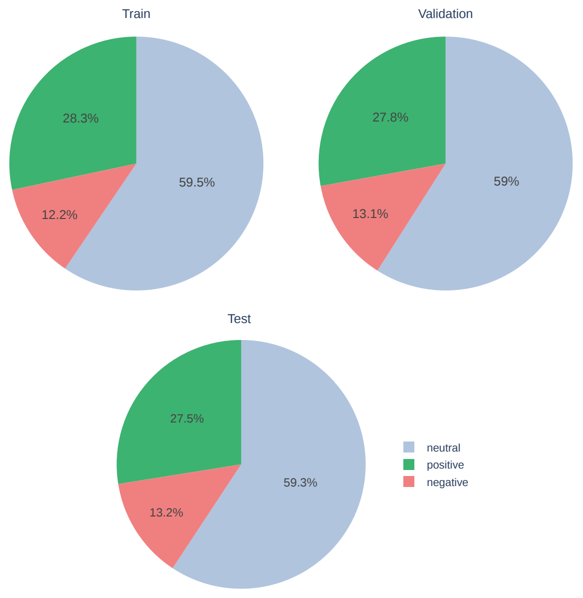


Figure 2: Train / Validation / Test Label Distributions

4.2. Evaluation Metrics

To evaluate the performance of the various sentiment classification models I trained, I report three metrics: accuracy, micro and macro F1 average. Macro F1 average treats all classes equally as it computes the metric independently for each class and then take the average. Micro F1 average will aggregate the contributions of all classes to compute the average F1 score. In a multi-class classification setup, micro average is preferable in case of class imbalance. Since the Financial PhraseBank suffers from label imbalance (almost 60% of all sentences are neutral), I will base the comparison of models on this particular metric.

4.3. Text Pre-processing

Financial headlines, similar to other real world text data, are likely to be inconsistent, incomplete and contain errors. Hence, to prepare the data, I perform initial pre-processing, starting with tokenization. Once the text tokenized, I remove stopwords and punctuation, which are required for correct grammar but add little to sentiment or context analysis. I used the stopwords corpus from NLTK for stopwords removal. Eventully, I lemmatized (i.e. grouped together the inflected forms of a word so they can be analysed as a single item) sentences using the WordNet Lemmatizer. After this initial filtering, I obtain the distributions of the number of words per sentence for the training set and the validation set (Figure 3).

4.4. Text Representation

In order to be able to process text data with a computer, one first need to encode words in a numeric form. To date, there are two main families of methods mapping words to vectors of real numbers. Statistical models offer simple categorical encodings. Yet, they treat all words as independent entities and are unable to provide any notion of similarity between words. With deep learning encoders, word embeddings become contextualized: they capture complex relationships, efficiently encoding semantic information that might be relevant to the task at hand.

4.4.1 Statistical Models

A very basic way to numerically represent words is through the count vectorizing (CV) method (also known as one-hot encoding). It simply consists of assigning each word its index in the vocabulary of unique words present in a text, and produces sparse vectors the size of the vocabulary. In spite of being fast to compute and relatively robust, this approach has some drawbacks. Such a representation suffers from the inability to provide any meaningful notion of similarity between words: in the resulting vector space, each word vector is orthogonal and equidistant to the others. By simply relying on counting, it also gives more weight to general, frequent words such as “the”, “and”, or “but”, which barely add meaningful information. As a result, important text features tend to vanish.

Term frequency - inverse document frequency (TF-IDF) is another statistical algorithm which addresses the feature-vanishing issue of count vectorizing algorithms. By re-weighting the count frequencies of words according to their number of appearances in the text, frequent words are given smaller significance.

Both of these methods suffer from the inability to capture the contextual information of a sentence. To overcome this, researchers have developed neural probabilistic models to convert discrete words into high-dimensional dense vectors capable of encoding the rich semantic information contained in text data.

4.4.2 Word Encoders

Sentiment analysis requires detailed semantic knowledge that is not provided by the statistical features extracted with

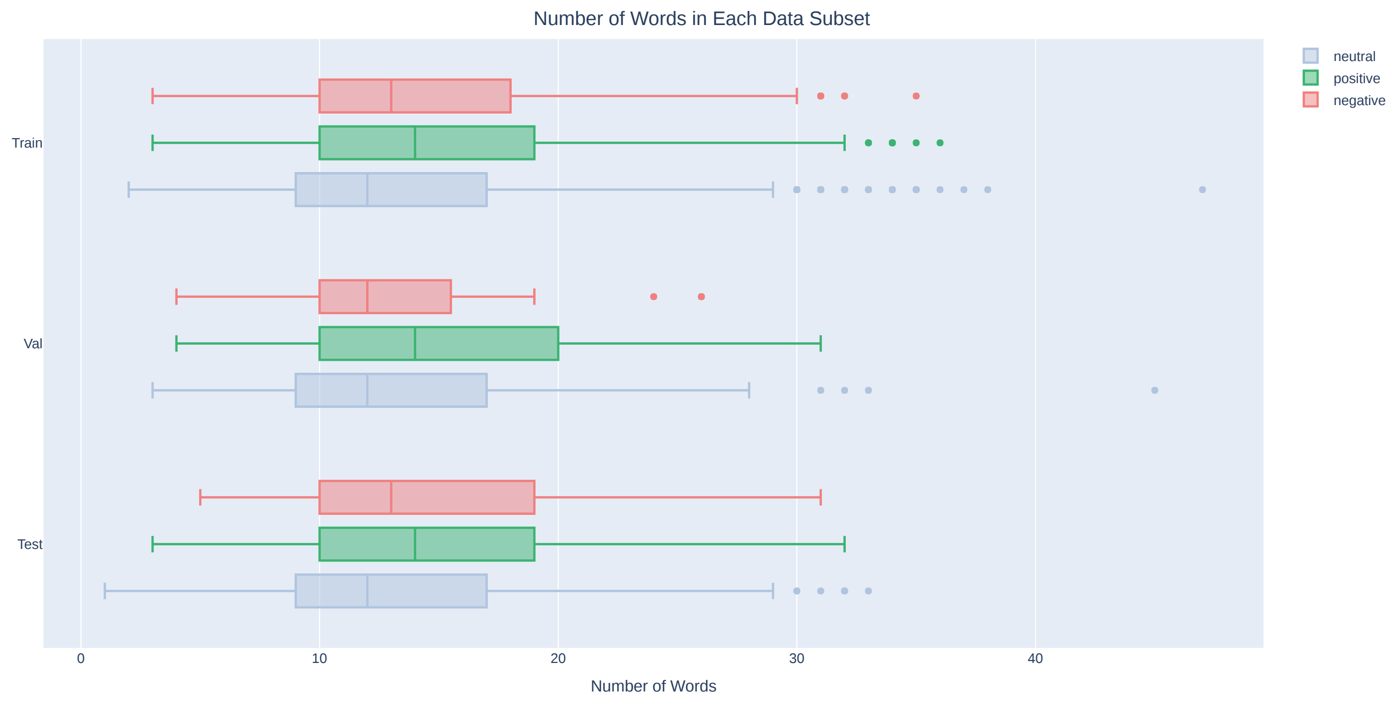


Figure 3: Boxplot of the Number of Words in Train / Validation / Test Subsets

CV and TF-IDF. Based on the distributional hypothesis, which assume that the complete meaning of a word is always contextual, researchers came up with word encoders. Word encoders classify words appearing in the same context as semantically similar to one another, hence assigning similar vectors to them. This approach, based on the principle famously summarised by J.R. Firth as "*you shall know a word by the company it keeps*", establishes a new area of research called distributional semantics. Let's present three of the most popular word emcoding techniques, namely Word2Vec [25], GloVe [26] and ELMo [27].

Word2Vec [25], introduced in 2013 by a team of researchers at Google, is considered to be the staring point of the still ongoing revolution in NLP. It computes continuous vector representations of words using two model architectures: Continuous Bag-of-Words (CBOW) and Skip-Gram (SG). The CBOW architecture predicts a center word from its (bag of) context words, while the SG architecture predicts context words given a center word. The authors show the effectiveness of both word encoding methods experimentally using several NLP applications, including sentiment analysis.

GloVe [26] was introduced in 2014 by a team of researchers at Stanford University. Based on a solid mathematical approach, GloVe overcomes the drawbacks of Word2Vec, ultimately improving the generated word embeddings. It emphasizes the importance of considering the co-occurrence probabilities between the words rather than single word occurrence probabilities themselves. GloVe is widely used as a word encoder for sentiment analysis.

Word2Vec and GloVe produce static embeddings for each words. This one-to-one mapping is, in many cases, inappropriatem qs words often have different meaning depending on the sentence they are used in. To account for this, a team of researchers at the Allen Institute introduced in 2018 an advanced word encoder called ELMo (Embeddings from Language Models) [27]. Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding. This is achieved leveraging a deep bidirectional language model, pre-trained on large corpora of textual data. The resulting encoder produces *contextual* word embeddings based on the whole context in which a word is used.

Transformers [2], introduced in 2018, build on top of a number of these clever ideas. The release of BERT (Bidirectional Encoder Representations from Transformers) [30], an event described as marking the beginning of a new era in natural language processing, brought efficient word encoding to another level. I give more details on the inner working of Transformers in Section 5.5.

5. Results

5.1. Baseline Method

The idea was to have reference results against which I could compare the performances of the more sophisticated models I would later develop. I decided to go with a simple baseline approach: fit a logistic regression on the one-hot encoded sentences. I also trained two other logistic regressions, one using GloVe embeddings (100-dimensional vectors), and another one using the embeddings extracted with a pre-trained version of BERT (768-dimensional vectors). In each experiment, the sentence embeddings simply consists of the sum of its word embeddings (Table 1). Interestingly, the one-hot encoded vectors seemed to produce better results. I haven't found an explaination to this observation.

Input	Accuracy	macro F1	micro F1
1-hot encodings	0.76	0.69	**0.76**
GloVe embeddings	0.59	0.25	0.44
BERT embeddings	0.72	0.68	0.72

Table 1: Baseline Results

5.2. Recurent Neural Networks

Recurent Neural Networks (RNNs) [31] are one of the primary architectures when it comes to dealing with sequencial data, such as text (a sentence is simply a sequence of words). They are particularly convenient as they can process inputs of any length and their size doesn't increase for longer inputs. Through the use of hidden states and shared weights, they were designed such that the computation for step *t* can use information from many steps back (at least in theory). In practive, however, they suffer from long term dependencies issues: the network has difficulties accessing information many steps back. This usually causes the gradient to vanish when backpropagating through earlier layers and the network to "forget" about past information. As such, RNNs are better at learning sequential recency than syntactic recency.

Table 2 summarizes the results obained when training RNNs in four different configurations. I first trained a plain 1-layer deep RNN on GloVe embeddings. It achieved 0.60 micro F1 average on the held out test set. Then, I trained a bidirectional RNN. It simply consists of putting two independent RNNs together: one is fed the input sequence in normal time order, while the other one is fed the same sequence in reverse time order. Both RNNs produce a sequence of hidden states, which are concatenated input-wise: concatenated hidden states rely on both left and right context. This is particularly valuable when trying to do sentiment analysis. Indeed, consider the following sentence:

The company released its latest results: these were terribly exciting !

We can regard the hidden state corresponding to the word *terribly* as a representation of this word in the context of this sentence. In the case of a plain RNN, this contextualized representation only contains information from the *left* context. Yet, the *right* context, which is the word *exciting*, modifies the true meaning of the word *terrible*, from negative to positive. This motivates why we might want to have information from both left and right when computing the contextualized representations of words in a sentence, which is precisely the purpose of bidirectional RNNs. Switching from unidirectional to bidirectional slightly improves the miro F1 average (from 0.60 to 0.62). The two other configurations simply consists of stacking 3 RNNs (be them plain or bidirectionnal) on top of each other. The best results were achived when training the 3-layer deep, bidirectional RNN

Model	Accuracy	macro F1	micro F1
RNN	0.67	0.47	0.60
bidir. RNN	0.65	0.50	0.62
stacked RNN	0.67	0.44	0.61
stacked bidir. RNN	0.68	0.53	**0.64**

Table 2: RNN Results

Several extensions of the classic RNN architecture have been designed to increase the memory capacity of the network along with the features extraction capacity. I will explore two of them: long short-term memory (LSTM) [32], and gated recurent units (GRU) [33].

5.3. Long Short-Term Memory

LSTMs are a type of recurrent neural network that allows long-term dependencies in a sequence to persist in the network by using "forget" and "update" gates. At each step *t*, we do not only have a hidden state h_t, but also a cell state c_t, which stores long-term information. LSTMs can erase, write and read information from the cell: the selection of which information is erased / written / read is controlled by three corresponding dynamic[2] gates. Mathematically, the cell can be seen as a route allowing the gradient to flow through the network without vanishing.

[2]dynamic means their value is computed based on the current context

Table 3 summarizes the results obained when training LSTMs in four different configurations, each time feeding GloVe embeddings. Here again, there is no significant difference between each configuration, though the 3-layer deep, bidirectionnal LSTM performs best.

Model	Accuracy	macro F1	micro F1
LSTM	0.76	0.71	0.76
bidir. LSTM	0.75	0.72	0.76
stacked LSTM	0.76	0.71	0.76
stacked bidir. LSTM	0.77	0.73	**0.77**

Table 3: LSTM Results

5.4. Gated Recurent Units

GRUs are a simpler alternative to LSTMs, as there is no cell state involved. This architecture introduces an update gate which determines the quantity of information to keep from the past as well as a reset gate which sets the quantity of information to forget. Having fewer parameters, they are faster to train. Table 4 summarizes the results obained when training GRUs in four different configurations, feeding GloVe embeddings. Once again, there is no significant difference between each configuration, though the 3-layer deep, bidirectionnal GRU performs best.

Model	Accuracy	macro F1	micro F1
GRU	0.77	0.74	0.77
bidir. GRU	0.77	0.73	0.77
stacked GRU	0.78	0.73	0.77
stacked bidir. GRU	0.78	0.75	**0.78**

Table 4: GRU Results

In spite of solving the vanishing gradient issue, LSTMs and GRUs still suffer from a major drawback: recurrent computation is slow. The sequential nature of these architectures precludes parallelization within training examples which becomes critical at longer sequence length as memory constraints limit batching across examples. Transformers [2] were introduced with the purpose to avoid recursion in order to allow parallel computation, as well as to reduce the observed drop in performances due to long-term dependencies.

Remark: I wish I had had time to use ELMo embeddings as well to assess the impact switching from static to dynamic word embeddings has on model's performances, but I didn't have time for this.

5.5. Transformers

The Transformer [2] is an attention-based architecture for modeling sequential information, that is an alternative to recurrent neural networks. It was proposed as a sequence-to-sequence model, therefore including encoder and decoder mechanisms. When performing sentence classification such as sentiment analysis, only on the encoder part is used. As clearly explained in [28], "*the encoder consists of multiple identical Transformer layers. Each layer has a multi-headed self-attention layer and a fully connected feed-forward network. For one self-attention layer, three mappings from* [token] *embeddings (key, query and value) are learned. Using each token's key and all tokens' query vectors, a similarity score is calculated with dot product. These scores are used to weight the value vectors to arrive at the new representation of the token. With the multi-headed self-attention, these layers are concatenated together, so that the sequence can be evaluated from varying "perspectives". Then the resulted vectors go through fully connected networks with shared parameters.*"

In 2018, researchers leveraged the transformer architecture to introduce a revolutionary language representation model, called BERT (Bidirectional Encoder Representations from Transformers) [30]. BERT relies on the unsupervised learning approach to pre-train deep bidirectional representations from large unlabeled text corpora by using two new pre-training objectives: 1) masked language modelling, which consists of predicting randomly masked words in a sentence, and 2) next sentence prediction, which consist of predicting whether two given sentences actually follow each other. In both case, the input sequence is represented with token and positional embeddings, the later being used to remember the order of words in the sequence. Two speicial tokens denoted by [CLS] and [SEP] are added to the beginning and end of the sequence respectively.

The most straight-forward way to use BERT is to use it to classify a single piece of text, which is precisely the ultimate goal of sentiment analysis. Table 5 shows the results obtained when training BERT from scratch vs. when fine-tuning a pre-trained version of BERT on the Financial PhraseBank dataset.

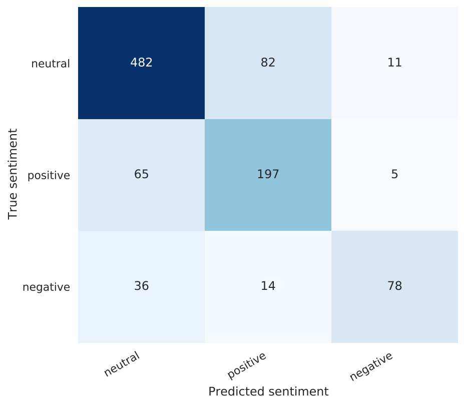
Configuration	Accuracy	macro F1	micro F1
BERT (from scratch)	0.67	0.55	0.66
BERT (pre-trained)	0.85	0.83	**0.85**

Table 5: BERT Results

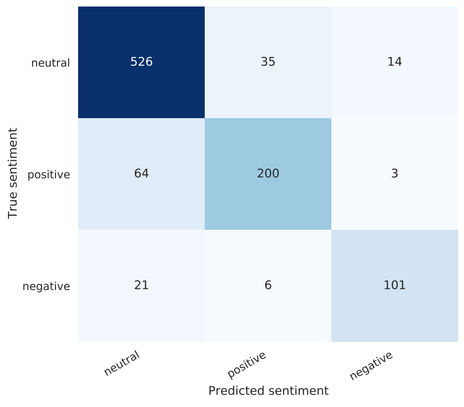
These results show the effectiveness of pre-trained language models, as the pre-trained BERT beats all previous models by a large margin. The BERT model trained from scratch performs poorly due to the relatively small size of the dataset I am using and the important number of parameters that such a model has to learn.

5.6. Error Analysis

Analysing where a given model successfully classsifies sentiments or fails to do so can reveal important information, either about the overall behavior – or tendency – of this model, or more general information such as the quality of the labels in the dataset. Figure 4 shows the confusion matrices of the best performing GRU and BERT models. Even though BERT is more accurate, we see that in both case, the main source of error comes from the positive vs. neutral sentences. Explicitely looking at some of the sentences where both models failed to predict the true sentiment might give some clues about the reasons behind these mistakes.



(a) GRU



(b) BERT

Figure 4: Confusion Matrices of the Best Performing GRU (a) and BERT (b) Models on the Held-out Test Set

6. Conclusion

This project presents a comprehensive study of NLP-based methods for sentiment analysis in finance. It begins with a word counting based approach, explores word encoders and concludes with Transformers, which are at the core of the on-going revolution in the domain. Transformers show superior performances compared to the other evaluated approaches. An important take away is that text representation methods, which feed the semantic meaning of words and sentences into the models, play a central role in sentiment analysis.

The financial domain is characterized by a domain-specific language, which adds to the difficulty of financial sentiment analysis. This aspect, which is not covered in this project, is an important topic of research for anyone looking to get improved performances. Even though generic (pre-trained) sentiment analysis models produce decent results, they probably lose some of their effectiveness when applied to specific domains such as finance. Further pre-training on a financial corpus (e.g. in the masked language model setting) and fine-tuning for sentiment analysis, as explored in [28], could lead to improved results.

Being completely new to natual language processing, this project was an opportunity to get familiar with the various popular methods used to process sequential data. With hindsight, I have developed a good understanding of some of the key concepts used in NLP research nowadays.

References

[1] Marc Velay and Fabrice Daniel. Using NLP on news headlines to predict index trends. 2018. 1

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. 1, 4, 6

[3] Burton G. Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, March 2003. 1

[4] Bing Liu. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015. 1

[5] M. Day and C. Lee. Deep learning for financial sentiment analysis on finance news providers. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1127–1134, 2016. 2

[6] Lodi Dodevska, Viktor Petreski, Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Lubomir Chitkushev, and Dimitar Trajanov. Predicting companies stock price direction by using sentiment analysis of news articles. 06 2019. 2

[7] Wataru Souma, Irena Vodenska, and Hideaki Aoyama. Enhanced news sentiment analysis using deep learning methods. *Journal of Computational Social Science*, 2, 01 2019. 2

[8] Sven Crone and Christian Koeppel. Predicting exchange rates with sentiment indicators: An empirical evaluation using text mining and multilayer perceptrons. *IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, Proceedings (CIFEr)*, pages 114–121, 10 2014. 2

[9] Chester Curme, H. Stanley, and Irena Vodenska. Coupled network approach to predictability of financial market returns and news sentiments. *International Journal of Theoretical and Applied Finance*, 18:1550043, 10 2015. 2

[10] Abinash Tripathy, Ankit Agrawal, and Santanu Rath. Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, 57, 03 2016. 2

[11] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. 11 2005. 2

[12] Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. 01 2009. 2

[13] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883, 2018. 2

[14] Dario Stojanovski, Gjorgji Strezoski, Gjorgji Madjarov, and Ivica Dimitrovski. Twitter sentiment analysis using deep convolutional neural network. volume 9121, 06 2015. 2

[15] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77, 02 2017. 2

[16] Tim Loughran and Bill Mcdonald. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66:35 – 65, 02 2011. 2

[17] Gang Wang, Tianyi Wang, Bolun Wang, Divya Sambasivan, Zengbin Zhang, Haitao Zheng, and Ben Zhao. Crowds on wall street: Extracting value from social investing platforms. 06 2014. 2

[18] Mathias Kraus and Stefan Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. *CoRR*, abs/1710.03954, 2017. 2

[19] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. pages 1422–1432, 01 2015. 2

[20] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015. 2

[21] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. 2

[22] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. pages 562–570, 01 2017. 2

[23] Sahar Sohangir, Dingding Wang, Anna Pomeranets, and Taghi Khoshgoftaar. Big data: Deep learning for financial sentiment analysis. *Journal of Big Data*, 5, 01 2018. 2

[24] M. Maia, A. Freitas, and S. Handschuh. Finsslx: A sentiment analysis model for the financial domain using text simplification. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 318–319, 2018. 2

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. 2013. 2, 4

[26] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014. 2, 4

[27] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2018. 2, 4

[28] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. 2019. 2, 3, 6, 7

[29] Pekka Malo, Ankur Sinha, Pyry Takala, Pekka Korhonen, and Jyrki Wallenius. Good debt or bad debt: Detecting semantic orientations in economic texts, 2013. 2

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. 2018. 4, 6

[31] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning internal representations by error propagation. 1986. 5

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 5

[33] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. 5