Classification of Financial Reports using Collaborative Topic Modelling

Gustav Kjellberg **Björn Hansson** **Felix Luthman** **Diar Sabri**

Abstract

In this report, we investigate the area of classification of financial reports. More specifically, we investigate how well financial reports can be used for predicting ratings, i.e the labels Buy, Sell, Hold. We employ a method presented by Cong et. al. in their paper *Textual Factors: A Scalable, Interpretable, and Data-driven Approach to Analysing Unstructured Information* [3]. The key idea is to 1) create a word embedding for each word in the corpus using Word2Vec. 2) Hash the word embedding using state-of-the-art Locality Sensitive Hashing. 3) group the words into clusters using the hashes and 4) engineer a feature vector for each document that is a representation of each cluster's relative importance for that document. We compare the accuracy of the model to a vanilla One-Hot encoding approach to evaluate if we are able to retain a sufficient amount of information using the clusters. The financial domain implies a financial specific vocabulary, therefore we also compare building Word2Vec word embeddings with only financial data, to using a pre-trained model provided by Google for transfer learning. We find that the method proposed by Cong et. al. does not suit the problem well enough to perform as good as, or better than, vanilla One-Hot encoding. Further, we also show that training Word2Vec on a strict financial corpus outperforms using Google's pre-trained model.

Project report for DD2430

Acknowledgements

We would like to express our gratitude to SEB for providing us with this interesting topic and the data required to conduct the research. Especially we would like to thank Ehsan Olfat for his continuous support and guidance throughout the project. Also, big thanks to Salla Franzén and Oscar Blomkvist for taking their time to share insights and knowledge about the specific task.

Contents

1 Introduction

Investigating the usefulness of data science to predict market trends has long been on the agenda, showing varying degrees of success. Usually, when determining market trends, analysts perform technical analysis of markets or companies to determine their growth potential. This information is then composed into financial reports and used as a basis for investment (i.e. buy, sell, hold). This has led to the accumulation of a large number of financial reports over time, which makes it possible to observe the market trends after the reports were published and whether or not the predictions were accurate.

This presents the opportunity of classifying the financial reports post-publication and using this information when evaluating new financial reports. While this may sound easy in theory, analysing large quantities of text data has historically proven to be difficult, especially since intricate information contained in text is often lost when the data is processed for a training process [3].

In this report, we investigate how well-suited financial reports are for classification using Natural Language Processing (NLP). We begin by creating a word embedding vector for each individual word, as opposed to simple One-Hot encoding. To do this, we use the Word2Vec methods which are available through the Gensim library. We then create clusters of words by using the hashing technique Locality Sensitive Hashing. The clusters aim to represent the similarities between different words by using their respective word embedding vectors as a basis for the hashing. In order to classify the reports with regards to their rating; (buy, sell, hold), we first create a feature vector by utilising the clusters. This is done with a topic modelling technique from which we are able to extract a feature vector for each document [3]. In short, the feature vector consists of the presence of each cluster within a certain report. The feature vector is then used with a simple classifier (Support Vector Machine and Random Forest).

The vocabulary at hand is biased towards finance and thus may introduce poor generalisation, especially with regards to common words that are not financial terms. Therefore, we further investigate the difference between (i) training the word embedding model Word2Vec from scratch to (ii) utilising a pre-trained model. The pre-trained model made available by Google has been trained on a large data set of English news articles. This could help with getting a more accurate representation of common words while we still introduce and train on finance specific terms.

1.1 Theoretical Background

1.1.1 Word2Vec

Word2Vec is a NLP model which utilise neural network technology for learning word associations. Once trained, the model can for instance be used to find synonyms and for creating linguistic contexts. The model takes a large amount of text data as input and produces a vector space as output, where each word from the input corresponds to a vector that describes the word's contextual position in relation to the other words in the text corpus [6].

1.1.2 Locality Sensitive Hashing

The main idea behind Locality Sensitive Hashing (LSH) is to find similar (based on some distance measure d) pairs of items, so called candidate pairs, with the help of a family of F hash-functions. If we let $d_1 < d_2$ be two distances based on the distance measure d, then a family F of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every f in F the following holds:

1. If $d(x, y) \leq d_1$, then the probability that $f(x) = f(y)$ is at least p_1
2. If $d(x, y) \geq d_2$, then the probability that $f(x) = f(y)$ is at most p_2

The above essentially means that similar pairs will be hashed equally with a high probability, and thereby considered as a candidate pair, while dissimilar pairs instead rarely hash to the same value, and with that, are considered as a candidate with a low probability [8].

One example of a family F and distance metric d that fulfills the LSH-conditions, is F as random hyper-planes and d as the Cosine Distance, which is what we will use throughout this report [3].

$$\text{Cosine Distance: } d(w_i, w_j) = 1 - \arccos\left(\frac{\langle w_i, w_j \rangle}{||w_i|| \cdot ||w_i||}\right) \tag{1}$$

$$\text{Random Hyperplane: } h_v(w) = sgn(\langle v, w \rangle), \text{ for v sampled from unit sphere } S^{p-1} \tag{2}$$

1.1.3 Support Vector Machine

An SVM is a classic machine learning classification algorithm that can be used to assign labels to objects [7]. The mathematical concept of the SVM aims to find the optimal hyperplane. The optimal hyperplane can be defined as the hyperplane which maximises the margin between the data points and thus separates the classes with the widest margin. The kernel function of the mathematical concept allows for the SVM to separate data which might not be linearly separable, by increasing the dimensionality of the separating hyperplane.

1.2 Random Forest Classifier

The Random Forest Classifier builds upon classical decision trees. It is an ensemble learning algorithm that creates multiple decision trees and then use the trees output in an ensemble fashion. The algorithm is good for large data sets and is able to explain each feature's importance which is valuable to get a better understanding of the model and its results [5].

1.2.1 Principal Component Analysis

Principal component analysis (PCA) is an algorithm which reduces the dimensionality of the data while simultaneously maintaining the variation in the data [10]. PCA is often used in exploratory data analysis and to visualise higher dimensional data by reducing the dimensions down to two or three dimensions. The mathematical concept behind PCA achieves the dimensionality reduction by identifying directions (principal components) and maximising the variation in the data in that direction.

2 Related Work

In 2004, Antweiler and Frank [1] published an article investigating the value of information posted on stock message boards in order to predict the value of a stock. One objective of their work was to discard the hypothesis that the information in the messages is just noise and thus that it actually provides valuable information. For instance, they use a Naive Bayes classifier to predict the label of a certain message where the labels, like ours, are {buy, sell, hold}. They find that the algorithm outputs a similar distribution of labels to the distribution obtained by manually classifying the messages. Further, they conclude that the information in the messages is not only noise but instead do provide valuable information.

More recently, in 2019, Hoberg et. al [4] used a novel approach for detecting emerging risk in the financial sector. They used latent dirichlet allocation (LDA) and semantic vector analysis (SVA) - a Word2Vec approach. Cong et. al [3] used a similar approach to Hoberg et. al in 2019, also using Word2Vec. Instead of LDA, Cong et. al used the state-of-the-art hashing method Locality Sensitive Hashing (LSH), which was then used for topic modeling and creating feature vectors for a document with regards of a topic's (cluster's) importance for that document. Their paper shows that LSH outperforms LDA as well as that a Word2Vec embedding outperforms a simple One-Hot count-based approach. This paper aims to apply the methods of Cong et. al to a specific task in the financial domain, which is to classify financial analyses.

Finally, this paper also builds upon the work made by O. Blomkvist in 2019. Blomkvist used Word2Vec word embeddings in combination with an LSTM to perform sentiment analysis on financial documents to classify them as either positive or negative [2]. This work was made for Skandinaviska Enskilda Banken (SEB), which is the same firm and objective that we work with. Both Blomkvist and Cong et. al compared results obtained from using Google's pre-trained Word2Vec model with a model trained only on the available data set. While Blomkvist found that the non-Google model performed better, Cong et. al found the opposite, which forms an interesting basis for investigation [2, 3].

3 Data

We were provided with data from SEB consisting of 25618 financial reports. These reports often contain one or more text fields with natural text as well as financial key figures, graphs and a recommendation to either buy, sell or hold stocks in the company analysed. Fortunately, these documents were parsed into a database hence we did not need to extract information from a PDF or alike. The distribution of data with regards to the relevant labels was

Buy	Sell	Hold
13983	2326	8933

Table 1: Data distribution with regards to label

As we can see, the data is heavily skewed towards Buy, which without any attempt to even out would limit the performance of our classifier. Therefore, the data used for training was evenly balanced, more on this in section 4.1. As mentioned previously, we obtained pre-parsed data and were therefore able to choose what sections of the analyses to use. As the objective is to perform classification we are only interested in the text, not graphs or any other visual information. We also want to mention that there is a fourth label with id 0 that indicates that the report has non of the above labels in Table 1. These reports were discarded.

4 Method

The sections 4.3 to 4.6 were largely based on the methodology presented in the *Textual Factors*[3] paper mentioned earlier, as the main objective of this project was to replicate their results on our specific domain. Their methodology can be summarised as:

By extracting a combined vocabulary from a large set of documents, and then subsequently finding vector representations for the words present in said vocabulary, word-clusters containing words with similar meaning were created. These word-clusters, which we roughly will interpret as educated guesses of actual underlying topics for the different documents, were given relative importance against each other, based on their frequency in the original documents. Once each topic is given such an importance factor, it is possible to create a vector representation for a new document, as the document can be assigned a real-value for each topic, based on said topic's importance and frequency in the new document. The final result is a real-valued vector representing how important each of the topics are for the given document.

As an extension to the framework presented in *Textual Factors*[3] we also attempt to classify our documents by the labels Buy, Sell and Hold, by using the vector representations for new documents as the input layer to a classifier.

Each of the steps mentioned above will be described in further detail below.

4.1 Data pre-processing

The dataset required some pre-processing in order for us to be able to properly operate on it. We performed the following pre-processing steps:

We removed all the stop words from the text data, i.e. the most common English words such as "the" and "of".

Once the stop words were removed, we also tokenised numbers, company names, countries, currencies, continents, sub-regions, dates and percentages (with distinction between positive and negative) to prevent the model from interpreting these as their own, separate words. For example; Volvo and Saab were replaced with <company>.

Last but not least, we also removed the mentioning of the labels within the documents, that is the words BUY, SELL and HOLD were removed so that they would not make the classification biased.

4.2 Balancing data classes

Since the data classes were heavily biased towards the buy-rating, the data set had to be balanced to prevent the bias from being reflected in the classification. We did this by simply extracting the same number of data points for each label, which was constrained by the label containing the least number of samples (Sell, 2326). The selection of which samples that were discarded or chosen, was random.

4.3 Word2Vec

Our model depends on having word embeddings, i.e. real-valued vector representations for words that appear in our dataset. There are 2 different approaches to solving this problem, either you can use a pre-trained Word2Vec model, such as Google's pre-trained model [6], or you can train your own. We decided to try both and utilised a pre-existing library called Gensim Word2Vec [9]. For the transfer learning model, we intersect our own model with the vocabulary of Google's model, assign the common words with Google's word embeddings and then train on our vocabulary.

4.4 Locality Sensitive Hashing clustering

Based on the vector embeddings from the previous section, the next step in our process was to cluster similar words together. We did this with a heuristic based on LSH, which allowed us to retrieve candidates for similar words. This procedure essentially boils down to hashing the vector representation of all the words in the vocabulary multiple times with different hash-functions sampled from random hyperplanes as described in equation 2. If two words are hashed to the same value by the same hash-function, they are considered a candidate pair, and a check for actual similarity can be performed. With these candidates, merging of similar words, and by extension, clusters, are possible. From a bird's-eye view, the process is summarised as:

Algorithm 1: LSH clustering

```
cluster_vocab($vocab$, $cluster\_limit$) =
    $clusters = vocab$
    $num\_clusters = size(vocab)$
    while $num\_clusters > cluster\_limit$ do
        $candidates = find\_candidate\_pairs()$
        $best\_candidate = most\_similiar\_pair(candidates)$
        $clusters.remove(best\_candidate.first)$
        $clusters.remove(best\_candidate.second)$
        $clusters.add(merge(best\_candidate))$
        $num\_clusters = num\_clusters - 1$
    end
end
```

4.5 Topic Modelling

The clustering allows us to gather a set of topics that correspond to each cluster. By going through each of our data-points for each topic and, based on the frequency of the words contained within that topic (cluster), give both each topic and word a relative importance. The result is a scalar value for each of the topics corresponding to its importance, as well as a vector in of which each element corresponds to a word's relative importance for its topic [3].

$$F_i = \frac{1}{\mathbf{1}^T N_{S_i} \mathbf{1}} N_{S_i}^T \mathbf{1} \tag{3}$$

$$d_i = \frac{\langle \mathbf{1}^T N_{S_i}, F_i \rangle}{\langle F_i, F_i \rangle} \tag{4}$$

F_i in equation 3 gives us the relative importance of each word belonging to the topic (cluster) i, whereas d_i gives us the importance of topic i itself [3].

The N_{S_i} term used in equation 3 and 4 refers to the document-term submatrix, i.e. a $D \times S_i$ dimensional matrix, where D corresponds to the number of documents for the topic and S_i is the number of words in the ith cluster [3].

4.6 Beta Loading

From the results of the Topic Modelling we are able to convert a document into a real-valued vector by assigning a value for a topic based on its presence within the document. By doing this for each topic, we are given an estimation of each topic's importance for the specific document, and thereby a vector can be created.

Let D be a new document, $N^{(D)}$ its document-term representation (containing the words present in the model's vocabulary), S_i denote the support of the words present in cluster i and K the total number of topics (clusters), then the beta loading for topic i can be calculated by the projection

$$x_i^{(D)} := \frac{\langle N_{S_i}^{(D)}, F_i \rangle}{\langle F_i, F_i \rangle}, \tag{5}$$

followed by the vector-representation for D:

$$D = (x_1^{(D)}, ..., x_K^{(D)}) \tag{6}$$

4.7 Classifier

The beta loading retrieved from equation 6 can then be utilised as a basis for a classifier. We decided to try both a SVM and a Random forest for this task, and for validation we used 10-fold cross validation. We also used PCA to reduce the dimension of the data.

4.8 Model Selection

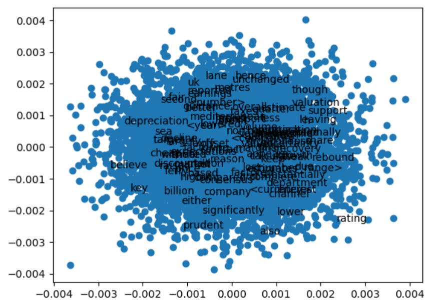
The scope of this project is to evaluate how well a financial textual document can be represented as a set of topics and the topics relative importance for that document. Choosing an optimal classifier has thus not been the main objective. Therefore, we choose to use a SVM to evaluate the performance of our models, we used both a radial and linear kernel to see if one was better than the other at separating the data. We also used a Random Forest classifier to get a better explanation of the model and to evaluate the feature importance.

4.9 Fine tuning

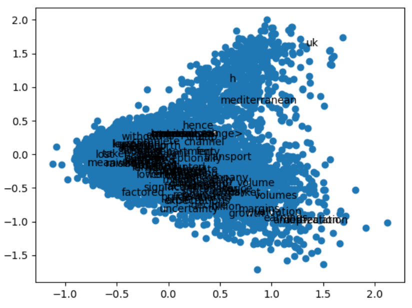
To improve the fairness of the clustering algorithm, we divided the β-value for each cluster c_i on the length c_i of itself. The reason for this was to reduce the impact of clusters of very large size. This seemed to improve the classification accuracy slightly. Furthermore, we tuned various hyper parameters for the Word2Vec model, such as window size and negative sampling values but also how many clusters that yielded the best accuracy.

5 Experiments & results

Our results indicate that the textual data do contain valuable information, such that we are able to classify the rating of an individual report with an average accuracy better than random. To study how much information is lost when translating a document of text into a lower dimension cluster representation, we use a vanilla One-Hot approach as a benchmark. The One-Hot encoded feature vector should be able to somewhat represent the documents information but is not able to include information such as the relation between words. We use this method as it should yield an accuracy that can be considered a bottom-line threshold for how much information that the documents contain. In total, we evaluate three methods, 1) Word2Vec trained only on our data, denoted (Own), 2) using transfer learning with Googles pre-trained Word2Vec model, denoted (Google), 3) using One-Hot encoding, denoted (One-Hot); with three different classifiers, SVM using a linear kernel, SVM using a radial kernel and Random Forest using 500 trees and no depth limit.

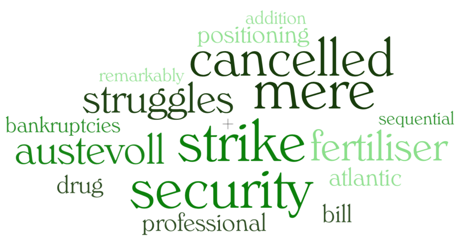


(a) Word embedding for words visualised for 2 dimensions using PCA and the model trained from scratch

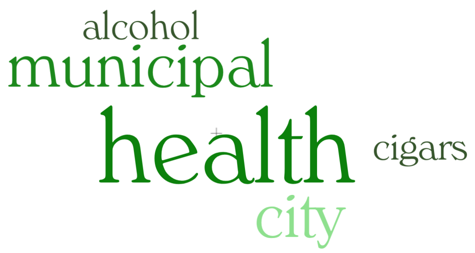


(b) Word embedding visualised for 2 dimensions using PCA and the model trained with Googles model

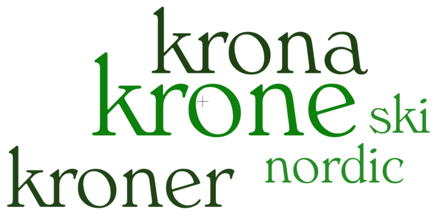
Figure 1: Visualisation of word embedding



(a) Word cluster example (own model)



(b) Word cluster example (Google model)



(c) Word cluster example (Google model)

Figure 2: Examples of generated clusters

Visualising the first words in the vocabulary of our two modes Own and Google, we see that the word embedding yields different results when reduced to 2 dimensions. Figure 1.a shows that the model Own yields a uniform cluster while figure 1.b, i.e the Google model, yields a different representation of the words. This indicates that there could be differences as their representation of the corpus is different. In table 2, 3 and 4 we show that this assumption is true. In figure 2 we see three clusters that the two word embedding approaches yielded. We can make the observation that the model which only uses the financial reports (figure 2 a) can be interpreted as finance specific. The two other clusters (figure 2 b, c), can instead be interpreted as more general. It is important to note that what the figure shows is a handmade selection of three clusters from the models, though through visual inspection of all clusters this pattern was representative of the two models.

We perform a 10-fold Cross-Validation (CV) for all our models, table 2, 3 and 4. We can see that the One-Hot encoded representation of documents out-perform the two other models with some margin. Actually, One-Hot yields better results in every fold and at worst it is better than the others are at their best. From further inspection of table 2, 3 and 4, we can see that the model trained from scratch outperforms the model built using transfer learning. We see that the worst fold for Own using SVMs is better than the best fold for Google using SVMs, while using Random Forest this does not hold but $\overline{Own_{Random\_Forest}} > \overline{Google_{Random\_Forest}}$ still do.

Fold	$Own_{svm-linear}$	$Google_{svm-linear}$	$One-Hot_{svm-linear}$
1	0.567	0.523	0.628
2	0.542	0.463	0.629
3	0.499	0.490	0.633
4	0.570	0.474	0.623
5	0.524	0.497	0.620
6	0.504	0.459	0.623
7	0.556	0.474	0.635
8	0.575	0.474	0.630
9	0.545	0.478	0.649
10	0.560	0.495	0.636
Average	0.544	0.483	0.630

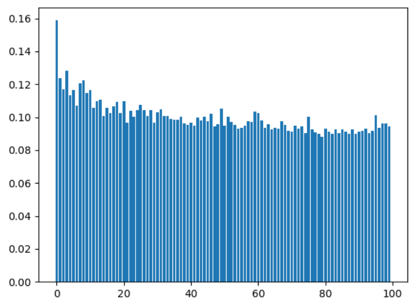
Table 2: 10-fold CV for SVM models with linear kernel

Fold	$Own_{svm-radial}$	$Google_{svm-radial}$	$One-Hot_{svm-radial}$
1	0.559	0.458	0.725
2	0.534	0.428	0.691
3	0.527	0.424	0.683
4	0.539	0.434	0.681
5	0.543	0.437	0.693
6	0.519	0.470	0.668
7	0.593	0.427	0.699
8	0.587	0.476	0.688
9	0.557	0.438	0.706
10	0.554	0.436	0.710
Average	0.551	0.443	0.694

Table 3: 10-fold CV for SVM models with radial kernel

Fold	$Own_{Random-Forest}$	$Google_{Random-Forest}$	$One-Hot_{Random-Forest}$
1	0.599	0.582	0.659
2	0.573	0.531	0.629
3	0.562	0.556	0.620
4	0.570	0.540	0.622
5	0.569	0.524	0.660
6	0.585	0.540	0.612
7	0.585	0.536	0.663
8	0.592	0.540	0.663
9	0.567	0.548	0.666
10	0.538	0.560	0.651
Average	0.573	0.546	0.645

Table 4: 10-fold CV for Random Forest models using 500 trees


(a) Word2Vec trained from scratch

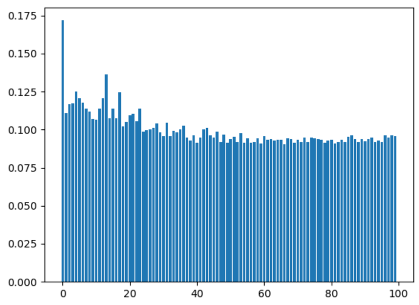

(b) Google's pretrained Word2Vec

Figure 3: RF feature importance of topic loadings (x: feature vector, y: feature importance)

As mentioned in section 4.8, we partly used a Random Forest classifier to get better explainability of the model and the feature importance. Figure 3 shows that for 100 features, the importance is evenly distributed. This implies that no feature is vastly more important than another and reducing the dimension further did not improve performance. In both figure 3 a and b the first feature is the most important for the model. Inspecting the cluster we saw that this was a token with format <token_name> that had a very high frequency in the data.

To prove that training from scratch outperforms the transfer learning approach we conduct a student's t-test. We evaluate h_0 = "Using Google's pre-trained model is better than training from scratch" using $P = 0.01$. With 1394 degrees of freedom, we can discard h_0 for all three models used in our tests. Thus we prove that using Google's pre-trained model is not better than training from scratch in this task, see table 5.

Model	t-value	critical value[1]
SVM_{linear}	16.39	2.326
SVM_{radial}	31.29	2.326
Random_Forest	9.99	2.326

Table 5: Statistical significance test comparing model Own to $Google$

Finally, we evaluate how well the Textual Factors frameworks is able to classify the rating compared to our baseline One-Hot approach. Table 6 shows the ratio between accuracy averaged over 10 runs using CV. We see that we are able to obtain approximately 89% of the baseline accuracy at best and 64% at worst. This indicates that we are not able to retain a sufficient amount of information when transforming documents to a cluster representation.

Comparison	Ratio
$(\frac{Own_{svm-linear}}{One-Hot_{svm-linear}})$	0.863
$(\frac{Googlesvm-linear}{One-Hot_{svm-linear}})$	0.766
$(\frac{Own_{svm-radial}}{One-Hot_{svm-radial}})$	0.794
$(\frac{Googlesvm-radial}{One-Hot_{svm-radial}})$	0.638
$(\frac{Own_{Random-Forest}}{One-Hot_{Random-Forest}})$	0.889
$(\frac{GoogleRandom-Forest}{One-Hot_{Random-Forest}})$	0.847

Table 6: Ratio between models and one-hot, given average score over 10 runs

6 Discussion

Judging by the resulting accuracy of the experiments, the topic loading of an article did not seem to be a good predictor of its rating. At best, we only reached a testing accuracy of around 57%, which is better than random but still significantly lower than the classification accuracy achieved with One-Hot vectors as the features. Regardless of the relatively good results yielded from One-Hot encoded feature vectors, we argue that it is not a viable option for the classification task. As mentioned in section 5, One-Hot encoding is not able to capture relation between words. Furthermore, the approach requires a feature vector of dimension V, where V is the number of unique words in the documents used for training.

There are many plausible reasons as to why the classification accuracy is so low. One reason could be that the pre-processing of the data was not sufficient. As mentioned in the textual factors paper [3], the efficiency of the approach was highly dependent on meticulous data pre-processing.

Another possible reason could be the choice of classifier and the selection of hyper parameters, since time constraints did not allow for a more thorough investigation in this area. For instance, a multi-layer perceptron with appropriately tuned hyper-parameters may have achieved a higher classification accuracy.

Other possible sources of errors include but are not limited to Word2Vec (maybe the BERT framework's word embedding would have worked better?) or simply that the unsupervised framework which was implemented can not be extrapolated to classification in this way. One big limitation of Word2Vec when used for document representation is that it only looks at words and not documents. Therefore, it might be infeasible to use it when working with trying to differentiate ratings in different documents. This can be seen by engineering a feature vector using the average of all word embedding in a document as this yielded results worse than random. As mentioned, we might see better results using state-of-the-art frameworks such as BERT or using Doc2Vec which is similar to Word2Vec but instead creates an embedding for entire documents.

6.1 Ethical considerations and sustainability

When it comes to classifying the rating of financial reports without any human intervention, it is important to consider that the research conducted in this report may lead to further digitisation of the financial industry. This may partly have a positive effect in the sense that could free up time by reducing tedious work for financial workers, but it is also possible that it could result in fewer job opportunities.

Sustainability wise, we argue that a digitisation of this area of work could potentially reduce the amount of resources needed (if it proves to be effective) in comparison with performing manual rating evaluation with human assistance.

Finally, we see no risk that this work could lead to misuse in any ethical aspect as the focus of the investigation has been on trying to represent textual data as a set of topics with regards to the topic's relative importance.

7 Conclusion

Our results show that creating the framework from scratch is significantly better than using transfer learning, likely due to the specific financial language used, which is coherent with Blomkvist's findings [2]. We also see that plain vanilla One-Hot encoding outperform the Textual Factors framework. As One-Hot encoding is one of the more trivial representations of a text document, we can conclude that the Textual Factors framework is not a viable option for classifying rating in financial reports.

8 Future work

This project has had a limited scope to investigate how well natural language in financial reports can be used as a basis for classifying rating. This implies that the focus has been on trying to, in the best of our capabilities, create the best possible word embedding and clustering. We have used a SVM classifier to benchmark our algorithms, to which we have given little attention. We believe that a good next step would be a two-fold approach of continuing the development of the overall framework and perhaps performing an ablation study on each separate component of the framework. The other part is to try and find a better choice of algorithm for classification. We have been discussing the potential of using a deep learning model which would hopefully serve the complexity of the data better. An example of such a model would be a Multi-Layer Perceptron which is a flexible ANN model and would perhaps yield interesting results. Another option would be to implement a gradient boosting technique, perhaps with the XGBoost library.

Also, due to the lack of financial domain knowledge within the group, we did not experiment with seed words during clustering[3]. As some financial terms may carry a lot of information that could benefit the clustering and hence the accuracy of the rating classification, we propose that this step is given greater thought in future work.

References

[1] W Antweiler and Z. M. Frank. Is all that talk just noise? the information content of internet stock message boards. The Journal of Finance, 2004.

[2] O. Blomkvist. Machine learning based sentiment classification of text, with application to equity research reports. 2019.

[3] W. L. Cong, T. Liang, and X. Zhang. Textual factors: A scalable, interpretable, and data-driven approach to analyzing unstructured information. 2019.

[4] Kathleen Weiss Hanley and Gerard Hoberg. Dynamic Interpretation of Emerging Risks in the Financial Sector. The Review of Financial Studies, 32(12):4543–4603, 02 2019.

[5] Adele Cutler Leo Breiman. Random forests.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[7] William S Noble. What is a support vector machine? Nature biotechnology, 24(12):1565–1567, 2006.

[8] Anand Rajaraman and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2011.

[9] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA.

[10] Markus Ringnér. What is principal component analysis? Nature biotechnology, 26(3):303–304, 2008.