Convolutional Neural Networks for Financial Text Regression

Neşat Dereli[†] and **Murat Saraçlar**[‡]
[†]Institute of Graduate Studies in Science and Engineering
[‡]Electrical and Electronics Engineering Department
Boğaziçi University, Istanbul, Turkey
nesat.dereli@boun.edu.tr, murat.saraclar@boun.edu.tr

Abstract

Forecasting financial volatility of a publicly-traded company from its annual reports has been previously defined as a text regression problem. Recent studies use a manually labeled lexicon to filter the annual reports by keeping sentiment words only. In order to remove the lexicon dependency without decreasing the performance, we replace bag-of-words model word features by word embedding vectors. Using word vectors increases the number of parameters. Considering the increase in number of parameters and excessive lengths of annual reports, a convolutional neural network model is proposed and transfer learning is applied. Experimental results show that the convolutional neural network model provides more accurate volatility predictions than lexicon based models.

1 Introduction

Most financial analysis methods and portfolio management techniques are based on risk classification and risk prediction. Stock return volatility is a solid indicator of the financial risk of a company. Therefore, forecasting stock return volatility successfully creates an invaluable advantage in financial analysis and portfolio management. While most of the studies are focusing on historical data and financial statements when predicting financial volatility of a company, some studies introduce new fields of information by analyzing soft information which is embedded in textual sources.

Kogan et al. (2009) defined the problem of forecasting financial volatility from annual reports as a text regression task and other studies contributed to the task because of its value (Wang et al., 2013; Tsai and Wang, 2014; Rekabsaz et al., 2017). There are also alternative soft information sources used for financial forecast like news (Tetlock et al., 2008; Nuij et al., 2014; Kazemian et al., 2014; Ding et al., 2015), online forums (Narayanan et al., 2009; Nguyen and Shirai, 2015), blogs (Bar-Haim et al., 2011) and bank reports (Nopp and Hanbury, 2015). However, annual reports are more informative and contain less noise since they are regulated by the government. On the other hand, annual reports are not suitable for short-term forecasting.

Volatility prediction using annual reports of companies is also a proper test-bed for natural language processing (NLP) since both volatility data and annual report data are freely available and no manual labeling is needed. In U.S., annual report filings, known as 10-K reports, are mandated by the government in a strictly specified format.

Previous works focus on sentiment polarity while forecasting the volatility. Their models are built on top of a financial lexicon (Loughran and McDonald, 2011) and most improvements are obtained by expanding the lexicon. However, a manually created lexicon should be updated over time and the solutions, depending on the lexicon, are not persistent.

In this paper, we propose an artificial neural network (ANN) solution which does not use a lexicon or any other manually labeled source. The convolutional neural network (CNN) model is designed similar to Bitvai and Cohn (2015) and Kim (2014). Nonetheless, annual reports contain excessively long text compared to movie reviews and this results in a more difficult task. To overcome this difficulty, max-over-time pooling layer is replaced by local max-pooling layer and transfer learning is applied.

The rest of the paper is organized as follows. In Section 2, we defined the problem. Section 3 introduces the model and its architecture. The details of our experimental settings, the results of the experiments and the analyses are presented in Section 4. Our work is concluded in Section 5.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 331–337
Florence, Italy, July 28 - August 2, 2019.

2 Problem Definition

In this section, stock return volatility which is aimed to be predicted is defined. Later, the dataset which is used in this work is introduced. Finally, evaluation measures are described.

2.1 Stock Return Volatility

Stock return volatility is defined as the standard deviation of adjusted daily closing prices of a target stock over a period of time (Kogan et al., 2009; Tsai and Wang, 2014; Rekabsaz et al., 2017; Hacısalihzade, 2017). Let S_t be the adjusted closing stock price for the day t. Then, the stock return for the day t is $R_t = \frac{S_t}{S_{t-1}} - 1$. Stock return volatility $v_{[t-\tau,t]}$ for τ days is given as

$$v_{[t-\tau,t]} = \sqrt{\sum_{i=0}^{\tau} \frac{(R_{t-i} - \bar{R})^2}{\tau}}.$$

2.2 Dataset

In this work, the dataset from Tsai et al. (2016), which is published online[1], is used because it includes up-to-date years and has enough reports for each year. Note that the datasets shared by Kogan et al. (2009) and Rekabsaz et al. (2017) are different than the dataset shared by Tsai et al. (2016) even if the same year is compared since the number of reports differ from each other. Hence, a direct performance comparison is not meaningful.

The dataset from Tsai et al. (2016) includes 10-K reports available on the U.S. Security Exchange Commission (SEC) Electronic Data Gathering, Analysis and Retrieval (EDGAR) website[2]. Following previous works (Kogan et al., 2009; Wang et al., 2013; Tsai and Wang, 2014; Tsai et al., 2016), section 7, Management's Discussion and Analysis (MD&A) is used instead of the complete 10-K report.

The dataset includes a volatility value for each report of 12 months after the report is published. The volatility value in the dataset is the natural logarithm of stock return volatility and used as the prediction target. We checked randomly sampled reports from SEC EDGAR and calculated volatility values by using adjusted closing stock prices from Yahoo Finance[3]. Both were consistent with the dataset.

[1] https://clip.csie.org/10K/data
[2] https://www.sec.gov/edgar.shtml
[3] https://finance.yahoo.com

2.3 Evaluation

Mean Square Error (MSE) is chosen as the main evaluation metric which is calculated by

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where $y_i = \ln(v_i)$.

Spearman's rank correlation coefficient is a measure which is used to evaluate the ranking performance of a model. Real volatility values and predicted volatility values can be used to calculate Spearman's rank correlation coefficient. Each set contains samples which consist of a company identifier and the volatility value of the company. Spearman's rank correlation coefficient of two sets is equal to Pearson's correlation coefficient of the rankings of the sets. The rankings of a set can be generated by sorting the volatility values of the set in an ascending order and enumerating them. The rankings of a set contains samples which consist of a company identifier and a volatility rank of the company. Spearman's rank correlation coefficient of the sets X and Y can be calculated by

$$\rho_{X,Y} = \frac{\mathrm{cov}(rank_X, rank_Y)}{\sigma_{rank_X}\sigma_{rank_Y}}$$

where $rank_X$ and $rank_Y$ represent the rankings of the sets X and Y respectively.

In all experiments, MSE is used as the loss function which means each model tries to optimize MSE. On the other hand, Spearman's rank correlation coefficient is reported only to evaluate the ranking performance of different models.

3 Model

The architecture of the network is presented in Figure 1 which is similar to previous works using CNN for NLP (Collobert et al., 2011; Kim, 2014; Bitvai and Cohn, 2015). Before reports are fed into embedding layer, their lengths are fixed to *m* words and reports with less than *m* words are padded. The output matrix of the embedding layer, $E \in \mathbb{R}^{km}$, consists of *k*-dimensional word vectors where the unknown word vector is initialized randomly and the padding vector is initialized as zero vector. Each element of the word vector represents a feature of the word.

The convolution layer consist of different kernel sizes where each kernel size represents a different *n*-gram. Figure 1 shows tri-gram, four-gram

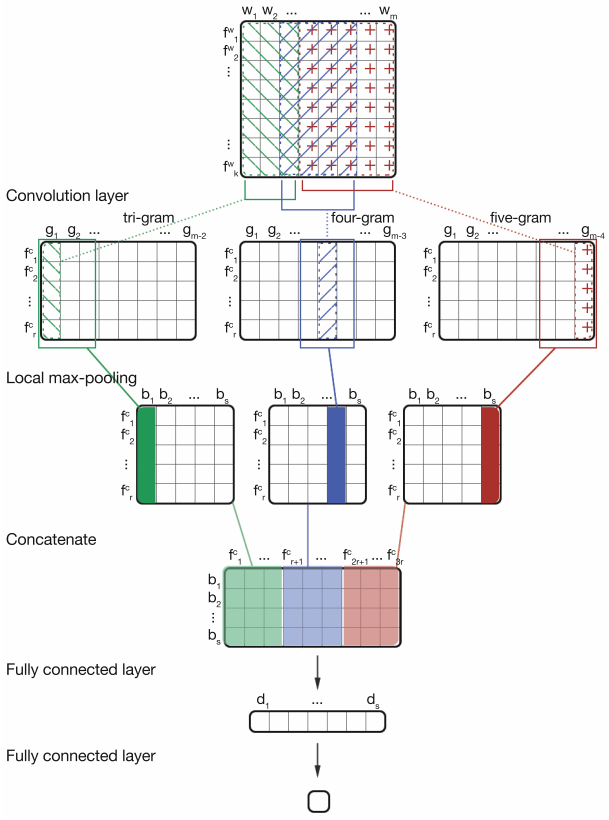

Figure 1: Network architecture of our baseline model (CNN-simple). A word embedded report through a single channel convolution layer with kernel sizes 3, 4 and 5 followed by a local max-pooling and two fully connected layers.

and five-gram examples. Let $n \in \mathbb{N}$ be the kernel width of a target *n*-gram. Each convolution feature $f_i^c \in \mathbb{R}^{m-n+1}$ is generated from a distinct kernel weight, $weight_i^n \in \mathbb{R}^{kn}$, and bias, $bias_i \in \mathbb{R}$. Rectified linear unit (ReLU) is used as the non-linear activation function at the output of the convolution layer,

$$f_{ij}^c = ReLU(weight_i^n \cdot w_{j:j+n-1} + bias_i).$$

Note that the convolution features, f_i^c have $m - n + 1$ dimension and they contain different information than word features, f_i^w. Convolution features are concatenated as

$$f_i^c = [f_{i1}^c, f_{i2}^c, ..., f_{i+n-1}^c].$$

g_i in Figure 1 represents each n-gram element thus there are $m - n + 1$ n-gram elements for each individual n-gram. Next step is local max-pooling layer which basically applies max-over-time pooling to smaller word sequence instead of the complete text (Le et al., 2018). Each sequence length is h and there are s outputs for each sequence,

$$b_i = \max(g_{ih:i(h+1)-1})$$

where $b_i \in \mathbb{R}^r$. After the local max-pooling layer is applied to all convolution layer output matrices, they are merged by concatenating feature vectors. Later, dropout is applied to the merged matrix and finally it is fed into two sequential fully connected layers. The presented neural network is implemented by using Pytorch[4] deep learning framework.

4 Experiments and Results

This section states preprocessing operations which are applied to the dataset. Pretrained word embeddings which are used in this work are described. Later, details of the setup of our experiments and the model variations are presented. Finally, the results of the experiments and the analysis of the results are discussed.

4.1 Preprocessing

MD&A section of the 10-K reports in the dataset are already tokenized by removing punctuation, replacing numerical values with # and downcasing the words. As in previous works, reports are stemmed by using the Porter stemmer (Porter, 1980), supported by Natural Language Toolkit (NLTK)[5]. Stemming decreases the vocabulary size of the word embeddings and thus reduces the parameters of the model. Stemming is also required to use word vectors trained by Tsai et al. (2016) since the corpora which is used to train the word embeddings consists of stemmed reports.

4.2 Word Embedding

Word embedding is a method, used to represent words with vectors to embed syntactic and semantic information. Instead of random initialization of the embedding layer of the model, initialization with pretrained word embeddings enables the model to capture contextual information faster and better. In our work, we used pretrained word embeddings supported by Tsai et al. (2016). They used MD&A section of 10-K reports from 1996 to 2013 to train the word embeddings with a vector dimension of 200 by word2vec[6] continuous bag-of-words (CBOW) (Mikolov et al., 2013).

[4]https://pytorch.org
[5]https://www.nltk.org
[6]https://code.google.com/archive/p/word2vec/

Model	2008	2009	2010	2011	2012	2013	Avg
EXP-SYN (Tsai 2016)	0.6537	**0.2387**	0.1514	0.1217	0.2290	0.1861	0.2634
CNN-simple (baseline)	**0.3716**	0.4708	**0.1471**	0.1312	0.2412	0.2871	0.2748
CNN-STC	0.5358	0.3575	0.3001	0.1215	0.2164	0.1497	0.2801
CNN-NTC-multichannel	0.5077	0.4353	0.1892	0.1605	0.2116	0.1268	0.2718
CNN-STC-multichannel	0.4121	0.4040	0.2428	0.1574	0.2082	0.1676	0.2653
CNN-NTC	0.4672	0.3169	0.2156	**0.1154**	**0.1944**	**0.1238**	**0.2388**

Table 1: Performance of different models, measured by Mean Square Error (MSE). **Boldface** shows the best result among presented models for the corresponding column.

4.3 Setup

The hyper-parameters of the CNN models are decided by testing them with our baseline CNN model. All weights of the baseline model are non-static and randomly initialized. Final hyper-parameters are selected as mini-batch size 10, fixed text length 20000, convolution layer kernels 3, 4 and 5 with 100 output features, probability of dropout layer 0.5, and learning rate 0.001.

Kogan et al. (2009) showed that using reports of the last two years for training performs better than using reports of the last 5 years. Rekabsaz et al. (2017) presented similarity heat-map of ten consecutive years and stated that groups consist of three to four consecutive years are highly similar. Our experiments also show that including reports which are four years older than test year into training set does not always help and sometimes even causes noise.

In this work, reports of three consecutive years were used for training while reports of the last year were used for validation to determine the best epoch. After the best epoch is determined, it is used as fixed epoch and the oldest year is ignored while the first step is repeated to train a new network without using validation set but fixed epoch instead. For example, reports of 2006 to 2008 are used as training set while reports of 2009 is used for validation. If the best result is achieved after 30 epochs, a new network is trained with reports of 2007 to 2009 through 30 fixed epochs. Finally, the trained network is tested for the year 2010.

Ignoring years older than four years prevent their noise effect but also reduces training set size. Experiments of this work show that old reports decrease the performance of the embedding layer but increase the performance of the convolution layer. The embedding layer can be biased easier than convolution layer since convolution layer learns features from larger structures (n-grams). Nonetheless, even training only the convolution layer using all years from 1996 to test year is time-consuming. Therefore, transfer learning is used by sharing the convolution layer weights which are trained on comparatively larger range of years. Yang et al. (2017) showed that relatedness of the transfer domains has a direct effect on the amount of improvement. Convolution layer weights are trained by freezing the embedding layer which is initialized with pretrained word embeddings and using years 1996 to 2006 for 120 epoch with early stopping. Other hyper-parameters are kept as described above.

4.4 Extended Models

Using transfer learning convolution layer, four different models are built. Since convolution layer weights are trained using pretrained word embeddings, those models perform well only when their embedding layers are initialized with pretrained word embeddings. Following Kim (2014), multichannel embedding layers are applied to some models.

- **CNN-STC:** A model with single channel non-static pretrained embedding layer and a transferred convolution layer which is static.
- **CNN-NTC:** Same as CNN-STC but its transferred convolution layer is non-static.
- **CNN-STC-multichannel:** A model with two channel of embedding layers, both are pretrained but one is static and other one is non-static. Transferred convolution layer is also static.
- **CNN-NTC-multichannel:** Same as CNN-STC-multichannel but its transferred convolution layer is non-static

Model	2008	2009	2010	2011	2012	2013	Avg
CNN-simple (baseline)	0.3884	0.0814	**0.5758**	0.5842	0.7064	0.7060	0.5070
CNN-STC	0.3875	**0.5226**	0.5570	0.5737	**0.7149**	**0.7341**	**0.5816**
CNN-NTC-multichannel	0.3727	0.4293	0.5187	0.5625	0.6531	0.7332	0.5449
CNN-STC-multichannel	0.3424	0.4042	0.4641	0.4924	0.4945	0.6305	0.4713
CNN-NTC	**0.3921**	0.4713	0.5500	**0.5910**	0.6978	0.7234	0.5709

Table 2: Ranking performance of different models, measured by Spearman's rank correlation coefficient. **Boldface** shows the best result among presented models for the corresponding column.

4.5 Results

Table 1 indicates that performance of our CNN-simple (baseline) model is comparable with EXP-SYN, the best model represented by Tsai et al. (2016), which uses a manually created lexicon and POS tagger. Furthermore, the best predictions for the years 2008 and 2010 are achieved by the CNN-simple model. Our best model, CNN-NTC, decreases the average error by 10% and produces the best predictions for the last three years of the experiment.

Ranking performance is valuable for some real world applications such as portfolio management. Furthermore, better ranking performance indicates better explanation of label distribution. Table 2 shows ranking performance of each model which is presented in this work. Spearman's rank correlation coefficient is bounded between -1 and 1. Higher Spearman's rank correlation coefficient means the model captures larger proportion of variability in the labels. It can be seen that ranking performance of CNN-NTC is as good as its regression performance. On the other hand, CNN-STC can model future distribution of stock return volatilities better than future values of stock return volatilities. It is important to note that our models use MSE as loss function and optimize MSE. Changing the loss function may improve ranking performance results and performance orders of the models.

4.6 Analysis

The embedding weights of CNN-NTC are compared with the pretrained word embeddings to determine the most changed words. While comparing the most changed word vectors, the words with yearly frequency less than 250 and more than 5000 are filtered out. Table 3 presents the top 10 most changed words and cosine distances to their pretrained vectors. Note that presented words are stemmed. Since words are in lowercase, the word *ETC* may cause confusion. It is an abbreviation and stands for Exchanged-Traded Commodity which is a common word in finance domain and stemmed version includes its plural form *ETCs* also. The stemmed words *concern*, *stabil* and *guaranti* are sentiment words and contained by finance sentiment lexicon (Loughran and McDonald, 2011). Having 3 sentiment words out of 10 words shows that our model uses sentiment information but not solely depend on sentiment words.

Word	Cosine Distance
anoth	0.2565
concern	0.2436
etc	0.2431
accordingli	0.2353
entir	0.2349
stabil	0.2328
increment	0.2308
thu	0.2306
situat	0.2167
guaranti	0.2120

Table 3: Top-10 most changed words, extracted from non-static embedding layer.

We also analyzed most changed sentiment word, *concern*, by extracting the 10 nearest words of pretrained word embeddings and CNN-NTC embedding weights separately (Table 4). It can be observed that *pertain*, *about* and *fear* are replaced with *safeti*, *trend* and *dmaa*. Stem words *safeti* and *trend* are related with the stem word *concern*. The word *pertain* is semantically very close to the word *concern*, they are even used interchangeably sometimes. However, *concern* can be replaced with *pertain* only if it does not have any sentiment polarity. It can be seen that expanding the lexicon using word embeddings, like previous works did (Tsai and Wang, 2014; Tsai et al., 2016; Rekabsaz et al., 2017), can be problematic and may end up with a lexicon expansion contain-

Static Embedding on 'concern'		Non-static Embedding on 'concern'	
Word	Cosine Distance	Word	Cosine Distance
regard	0.2772	regard	0.3233
privaci	0.5287	privaci	0.5433
inform	0.5587	safeti	0.5550
debat	0.5706	inform	0.5562
implic	0.5817	trend	0.5568
heighten	0.5825	heighten	0.5692
pertain	0.5844	inquiri	0.5959
about	0.5901	dmaa	0.6013
inquiri	0.5919	debat	0.6025
fear	0.5954	implic	0.6033

Table 4: Top-10 most similar words to *concern* comparing their word vectors.

ing semantically close but sentimentally far words.

Another interesting word in the list is *DMAA*. It stands for dimethylamylamine which is an energy-boosting dietary supplement. In 2012, the U.S. Food and Drug Administration (FDA) warned DMAA manufacturers. In 10-K report of Vitamin Shoppe, Inc. published on February 26, 2013, concern of the company about DMAA is stated:

> "If it is determined that **DMAA** does not comply with applicable regulatory and legislative requirements, we could be required to recall or remove from the market all products containing **DMAA** and we could become subject to lawsuits related to any alleged non-compliance, any of which recalls, removals or lawsuits could materially and adversely affect our business, financial condition and results of operations."

It shows that the CNN model focuses on correct word features but also can overfit easier. In financial text regression task, the word *DMAA* is quite related with the word concern but it is not a common word and also sector specific.

5 Conclusion

The previous studies depend on a financial sentiment lexicon which is initially created by manual work. This paper reduced both dependencies by using word vectors in the model. Word vectors are used in previous studies to expand the lexicon but they are not included to the model directly. On the contrary, our work includes word vectors directly to the model as main input.

In addition, transfer learning is applied to the convolution layer since effect of temporal information on distinct layers differs. Evolving word vectors are analyzed after model benchmarks. The analysis demonstrates that CNN model tracks sentiment polarity of the words successfully and it does not depend on sentiment words only. However, it is also observed that CNN models can overfit easier.

This work is focused on text source and did not include any historical market data or any other metadata. Further research on including metadata to CNN model for the same task may increase the value and analysis.

Acknowledgments

The numerical calculations reported in this paper were performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources). We would like to thank to Can Altay, Çağrı Aslanbaş, Ilgaz Çakın, Ömer Adıgüzel, Zeynep Yiyener and the anonymous reviewers for their feedback and suggestions.

References

Roy Bar-Haim, Elad Dinur, Ronen Feldman, Moshe Fresko, and Guy Goldstein. 2011. Identifying and following expert investors in stock microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1310–1319, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Zsolt Bitvai and Trevor Cohn. 2015. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 180–185, Beijing, China. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2327–2333. AAAI Press.

S.S. Hacısalihzade. 2017. *Control Engineering and Finance*, Lecture Notes in Control and Information Sciences, page 49. Springer International Publishing.

Siavash Kazemian, Shunan Zhao, and Gerald Penn. 2014. Evaluating sentiment analysis evaluation: A case study in securities trading. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 119–127, Baltimore, Maryland. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, Boulder, Colorado. Association for Computational Linguistics.

Hoa Le, Christophe Cerisara, and Alexandre Denis. 2018. Do convolutional networks need to be deep for text classification ?

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 180–189, Singapore. Association for Computational Linguistics.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1354–1364, Beijing, China. Association for Computational Linguistics.

Clemens Nopp and Allan Hanbury. 2015. Detecting risks in the banking system by sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 591–600, Lisbon, Portugal. Association for Computational Linguistics.

Wijnand Nuij, Viorel Milea, Frederik Hogenboom, Flavius Frasincar, and Uzay Kaymak. 2014. An automated framework for incorporating news into stock trading strategies. *IEEE Trans. on Knowl. and Data Eng.*, 26(4):823–835.

MF Porter. 1980. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14.

Navid Rekabsaz, Mihai Lupu, Artem Baklanov, Alexander Dür, Linda Andersson, and Allan Hanbury. 2017. Volatility prediction using financial disclosures sentiments with word embedding-based ir models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1712–1721, Vancouver, Canada. Association for Computational Linguistics.

Paul C. Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467.

Ming-Feng Tsai and Chuan-Ju Wang. 2014. Financial keyword expansion via continuous word vector representations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1453–1458, Doha, Qatar. Association for Computational Linguistics.

Ming-Feng Tsai, Chuan-Ju Wang, and Po-Chuan Chien. 2016. Discovering finance keywords via continuous-space language models. *ACM Trans. Manage. Inf. Syst.*, 7(3):7:1–7:17.

Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. 2013. Financial sentiment analysis for risk prediction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 802–808, Nagoya, Japan. Asian Federation of Natural Language Processing.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks.