Fortia1@the NTCIR-14 FinNum Task: Enriched Sequence Labeling for Numeral Classification

Abderrahim Ait Azzi[0000−0002−3094−6791] and Houda Bouamor[0000−0001−8605−957X]

Fortia Financial Solutions, 17 Avenue George V, 75008, Paris, France
firstname.lastname@fortia.fr

Abstract. This paper describes our submission to the NTCIR-14 - FinNum Shared Task on Fine-Grained Numeral Understanding in Financial Tweets. We participate in both *Subtask-1* and *Subtask-2*. We formulate the problem as a sequence labeling task and design a hybrid approach where we use external linguistic and non-linguistic features to enrich word level representation within a CNN-based neural network. Since the two subtasks are strongly related, for *Subtask-2*, we introduce a *fusion* approach in which our model considers information about the category predicted in *Subtask-1* when assigning a sub-category to each numeral. Our models achieve an F_1 score of 93.94% (micro) and 90.05% (macro) on Subtask-1 and 87.17% (micro) and 82.40% (macro) for Subtask-2 on the test set. This ranks us 1*st* in the competition in both Subtask-1 and Subtask-2.

Keywords: Numeral classification · Tweet analysis · Sequence labeling · Neural models · Enriched Embeddings · Linguistic features
Task Name: FinNum.
Subtasks: Subtask1 and Subtask2.

1 Introduction

The growing popularity of social media and user-created content is producing massive quantities of textual information. The popular micro-blogging service *Twitter* is one particularly fruitful source of such kind of user-created content.[1] Twitter has been playing an increasingly important role for individuals to share their own opinions on many finance-related topics and services. It provides a real-time information channel that includes not only major news stories but also minor events that, if properly analyzed, can provide valuable information about the market. This motivated a series of research work to introduce various technologies to advance understanding of the financial market dynamics [15] ranging from sentiment analysis [12] to credit risk evaluation [7] and stock market movement prediction [9, 14], where behavioral finance researchers can apply

[1] twitter.com

computational methods to large-scale Twitter data to better analyze, understand and predict markets.

In stock market prediction for example, numerals play an important role in forecasting the movement of asset prices based on the past market data. To predict the price trend, investors may use technical indicators calculated with history price or analyze price charts and look for the embedded patterns [1], [2].

The goal of the FinNum shared task is to leverage the numeric opinions made by the crowd on Twitter by understanding the meanings of numerals. The shared task aims at classifying numerals into 7 main categories, and then into 17 subcategories (See Table 1). A dataset of financial tweets containing numeric values is provided. This dataset is compiled from *StockTwits*, a microblogging platform exclusively dedicated to the stock market.[2]

In this paper, we present our submissions to this shared task: we participate in both *Subtask1* and *Subtask2* with an enriched CNN based models (E-CNN). We formulate the problem as a sequence labeling task and design a hybrid approach where we use external linguistic and non-linguistic features to enrich word level representation within our CNN-based neural network. We describe our models as well as our experimental setups and report the results obtained for both subtasks on the Development and Test sets. Our models are ranked first in both subtasks with a micro-averaged F_1 score exceeding 93% for Subtask-1.

2 FinNum Task Definition

FinNum shared task consists in classifying numerals in financial tweets into a set of predefined categories and subcategories. Two classification Subtasks are proposed: (i) **Subtask-1**, a 7-way classification task covering 7 numeral categories and (ii) **Subtask-2**, where the classification is extended to include 17 numeral subcategories.[3]

Table 1 shows Subtask-1 categories and their associated subcategories, as well as the distribution of each category and subcategory in the training dataset.

[2] https://stocktwits.com/

[3] Three categories (Indicator, Quantity, and Product/ Version number) do not have a subcategory. Thus, the category and subcategory information is the same for these three categories.

Fig. 1: Categories and subcategories in Subtasks 1 and 2, given along with their distribution in the FinNum dataset.

Category	%	**subcategory**	%	**Category**	%	**subcategory**	%
Monetary	35.96	money	8.58	**Percentage**	12.21	absolute	3.69
		quote	11.54			relative	8.53
		change	2.08	**Option**	2.46	exercise price	1.65
		buy price	4.65			maturity date	0.81
		sell price	1.50	**Indicator**	2.43		2.43
		forecast	2.08	**Temporal**	34.46	date	30.30
		stop loss	4.65			time	4.15
		support or resistance	3.29	**Quantity**	10.80		10.80
				Product	1.66		1.66

3 Data and Preprocessing

We trained and evaluated our systems for both subtasks using the data provided for the shared task, a set of 5,282 tweets where numerals are annotated with their categories and subcategories [1]. This set is divided into 4,072 tweets for training, 457 for development and 753 for testing.

We pre-processed the whole dataset by tokenizing each tweet using keras word tokenizer [4] while preserving the target numbers as they appear in the original tweet. We also remove all non_Ascii characters and replace URLs with the special token (_URL_). Contrarily to [15], we keep the tweet ids, hashtags, and cashtags in their original format, as they might be related to target numerals and contain indications about their categories. However, we remove emojis, because emojis could present the market sentiment of investors, but could not show the fine-grained opinion defined in the taxonomy provided in the shared task. Finally, all remaining tokens are transformed into lowercase.

4 Sequence Labeling for Numeral classification

Table 1: Example of input-output

Tweet	$DPW	calm	before	the	storm	.	For	**10**	am	est	is	**2.29**	check	back	tomorrow
Subtask-1	O	O	O	O	O	O	O	**Temporal**	O	O	O	**Monetary**	O	O	O
Subtask-2	O	O	O	O	O	O	O	**time**	O	O	O	**forecast**	O	O	O

We formulate our two classification subtasks as sequence labeling problems, where a label is assigned to each token in the tweet including target numbers: ***O*** for regular words, $\boldsymbol{C} \in$ ***Categories*** for each target number, in Subtask-1, and $\boldsymbol{c} \in$ ***subCategories*** for each target numeral in Subtask-2. For instance, in the tweet given in Figure 1, the category of the target numeral x_T =*10* is *Temporal* and its subcategory is *time*.

For both subtasks, we design and implement a neural architecture for sequence labeling. An overview of our model is depicted in Figure 2. For both subtasks, we use a Convolutional Neural Network (CNN) on word embeddings concatenated with an additional set of specific features, to detect the correct label for each token in a given tweet.

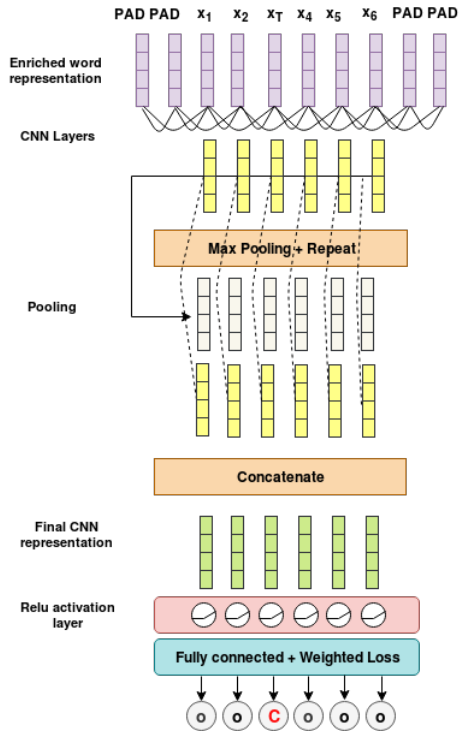


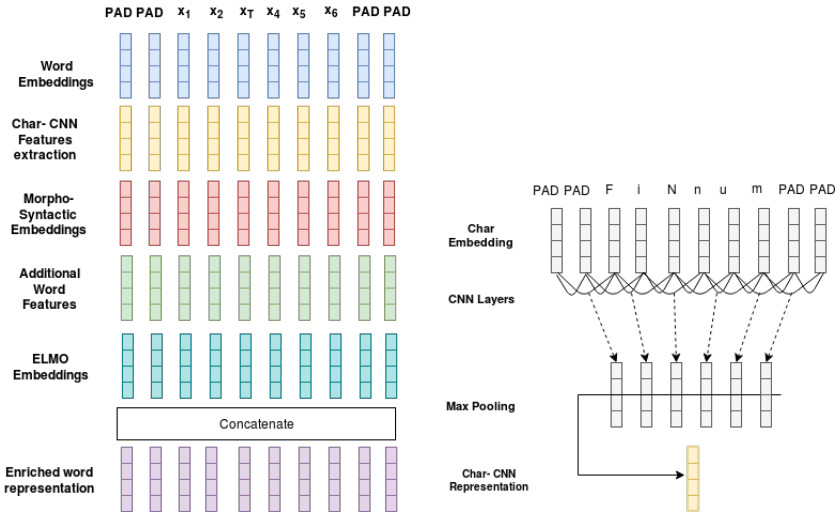
Fig. 2: Architecture of *E-CNN* Model

In the following, we describe each component in our neural network top to bottom.

Word-level Embeddings The first component is an embedding layer that converts every input sequence into sequences of low-dimension dense vectors via a lookup table.

CNN for Character-level Representation Previous studies have shown that using CNN is an effective method to extract morphological information (like the prefix or suffix of a word) from characters of words (*OOV* words included) and encode it into neural representations [3, 6]. This gives a rich representation of out-of-vocabulary words (instead of associating all *OOV* to the common UNK

token). In our dataset, due to the informal style of writing in tweets, many words in the development and test sets do not appear in the training set and thus are considered as OOVs.
Figure 3b illustrates the Char-CNN we use to extract the character-level representation of a given word. This CNN is similar to the one used in [3] except that we use only character level embeddings as the inputs to the CNN.



(a) Description of our enriched word representation

(b) The CNN used for extracting character-level representations of words.

Fig. 3: Enriched word representation

ELMo Word Embeddings In order to further enrich our word representation, we use pre-trained contextualized ELMo embeddings[4] [11] of dimension 1024. These word embeddings differ from other word embeddings such as GloVe [10] in that each token is represented by a vector that is a function of the whole sentence (a word can thus have different representations depending on the sentence it is extracted from). Furthermore, ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training. ELMo embeddings have been shown to give state-of-the-art results in multiple NLP tasks [11].

Morpho-syntactic Embeddings In general, morpho-syntactic information encoded in part of speech (POS) tags could be a good indicator of the category of a given token. For instance, in our scenario,using a POS tag could help in the

[4] https://allennlp.org/elmo

Fig. 4: Example of a tweet tagged using the ARK TweetNLP POS tagger

Tweet	$DPW	calm	before	the	storm	.	For	**10**	am	est	is	**2.29**	check	back	tomorrow
POS Tags	N	A	P	D	N	,	P	$	V	N	V	$	N	R	N

prediction of the right numeral category and sub-category like Quantity where the numeral is often followed with a noun.
To build this layer, each tweet in our dataset is tagged using the CMU ARK Twitter Part-of-Speech Tagger [8]. This toolkit is based on a CRF sequence labeling model and uses a set of 25 POS tags covering hashtags, at-mentions, and urls in addition to punctuation, nominals, verbs, and other open-class and closed-class words. An example of the output of this tagger on a Tweet from the training set is illustrated in Figure 4. Similarly to traditional word embeddings, we assign different trainable vectors for each part-of-speech tag.

Additional word-based features In sequence labeling, enriching word representation with hand-crafted features has been shown to improve models performance greatly [13]. Inspired by this, we combine our neural model with a set of features. For instance, we indicate if the word (or its neighbors) is composed only of alphabetic, contains non-alphabetic characters, upper case, lower case, number, or punctuation. Furthermore, we extend the list of keywords from [1], where each keyword is related to a list of tokens indicating a certain category or subcategory shown in Figure 5, by adding new keywords. We then check if a given word from the tweet appears in this list.

Fig. 5: Extended list of keywords per category feature

key_p	%,percent,pc
key_r	up, down , decline, increase, growth,loss, gain, lose, + ,- ,decreased,decreases
key_m	january, jan, february, feb, march, mar, april, apr, may, june, jun, july, jul, august, aug, september, sep, october, oct, november, nov, december, dec, /, th
key_i	ma, dma, sma, ema, rsi, ichimoku
key_d	day , week , month ,year,mos,yrs,/,days, age , ages , weeks , week , Q
key_t	second , seconds , minute , minutes , hour , hours , p.m. , s , a.m
key_o	calls , call , put , puts
key_s	sold , selling , sell
key_b	bought , buying , buy , cost

Convolutional Layer Once the five components described above are built, we concatenate their outputs to have an ***enriched*** word representation as illustrated in Figure 3a. Then, we use three 1-dimensional convolutional layers

applied in parallel to the word representation sequence, each using a different filter size k and performing a convolution operation with a ReLU activation in order to capture different n-gram features from the input sequence.
Note that we set the kernel size k to an odd number, and further pad the left and the right positions of the input sequence with all zeros in order to have each CNN-output well aligned with the original input sequence $x_1, \ldots, x_n$, as illustrated in Figure 2.
Our intuition behind using a CNN instead of a Bi-LSTM, here, is that local context can be sufficient to give an idea about the correct token label. Also, as Bi-LSTMs are computationally expensive compared to CNN, it is more practical to use them for solving such classification tasks.

Pooling When all the representations of words are calculated, we apply a max-pooling layer to generate a vector representation of the entire tweet. Then, we concatenate this tweet-level representation with each word-CNN representation. This way, the last layers consider also the entire tweet representation (the global context), when predicting each label.

Weighted cross-entropy The distribution given in Table 1 shows that the number of examples for some categories is significantly greater than that of the others (i.e., Monetary: 35.96% *vs.* Product: 1.66%). Also, the way we designed our sequence labeling task, increases the data imbalance, as the majority of the tokens in any tweet are labeled with the special tag **O** (there are more common words in a tweet than numerals). This can add some undesired bias to our predictions.
To tackle this challenge, one of the most straightforward ways is to introduce a weight for each class to put more emphasis on the less represented classes and find an acceptable trade-off between different categories. We use the following weighted cross-entropy as a loss function:

$$Loss = -\sum_{c=1}^{M} w_c y_{k,c} \log(p_{k,c})$$

Where M is the number of classes and y is binary indicator that indicates whether the assigned class label c for observation k is correct. p refers to the predicted probability of observation k being labeled with class c.
We define the weights using the following formula:

$$w_c = \frac{1}{\sum_{i=1}^{n} \mathbf{1}_{c(i)=c}}$$

The weights depend heavily on the number of occurrences of the classes in the training set (classes with low occurrences are assigned high weights).
Our final model combining all the components described above is called: an **E**nriched **C**onvolutional **N**eural **N**etwork, *E-CNN* henceforth.

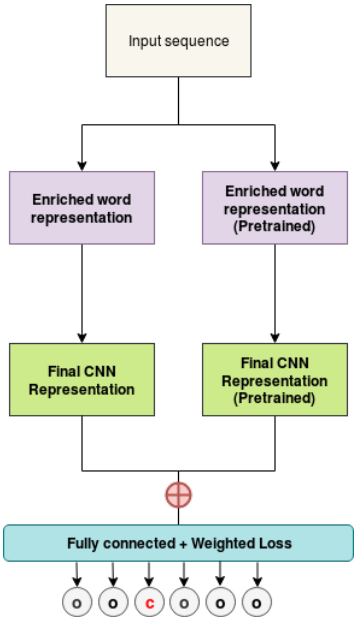


Fig. 6: Diagram of our *fusion model*, which uses a pre-trained model of a subtask 1 to train a second model for subtask 2. The separate CNN-representation are combined through concatenation.

5 Fusion model

In our experiments, both subtasks 1 and 2 are solved using the same *E-CNN* model. However, in order to take advantage of the model used to train and predict categories for Subtask-1, and use it to drive the quality of sub-category prediction in Subtask-2, as both subtasks are strongly related, we propose a fusion approach to encourage conditioning the prediction of sub-categories on the categories assigned in Subtask-1.

Once our first *E-CNN* model (*E-CNN-1*) is trained to predict numeral categories, we save the weights resulting from all the layers (including the enriched embeddings and the CNN representation). Then, we train a second *E-CNN* model that has access to the (fixed) pre-trained weights of the first model *E-CNN-1*.

We combine the two models using the fusion approach illustrated in Figure 6. This approach was explored in other NLP applications including text generation [5].

6 Experimental Setup

We present in this section our experimental settings and describe the tools used. We implement the two models for both subtasks using Keras[5] [4] with a tensorflow backend.
We train our models on the training set provided for each subtask with mini batches of size 32 and a default learning rate of 0.001. We use a word embedding size of 100, POS tag embedding size of 150 and CNN kernel sizes of 3, 5 and 7 and 60 CNN filters for word-CNN and 50 for char-CNN. We use a grid search algorithm to find a set of optimal parameters.

6.1 Training settings

For training, we use a $k-fold$ cross-validation (with $k = 15$). In this procedure, k different *E-CNN* models are trained on k different subsets of the training data. Each of the k models is used as a member of an ensemble.
We use a majority vote approach to predict the correct class: we store the prediction of each model for each target numeral, the class that receives the highest number of votes from the k models is kept as the final category or sub-category of the target.
We train each model for a total of 30 epochs with an early stopping patience of 3. Generally, the training stops after 6∼7 epochs for Subtask-1 and 11∼12 epochs for Subtask-2. Using a machine with a GPU (Nvidia GeForce GTX 1050 Ti Max-Q GPU 4GB), the training required only ∼1 minute per epoch, which made training multiple models very practical.

6.2 Inference settings

In the inference step, for Subtask-1, we assign to a given numeral the best category predicted by the model (different from the label **O**):

$$\mathbf{C}^* = \underset{C \in \mathbf{Cat} \ \&\ C \neq \mathbf{O}}{\operatorname{argmax}} \ p\ (C \mid \mathbf{x})$$

For Subtask-2, two strategies are plausible, we either choose the best label among the whole set of sub-categories **subCat** (formula 1), or use the prediction of Subtask-1 to select a label as a subcategory of the optimal category predicted in subtask 1 (formula 2):

$$\mathbf{c}^{*(1)} = \underset{c \ \in \mathbf{subCat} \ \&\ c \neq \mathbf{O}}{\operatorname{argmax}} \ p(c \mid \mathbf{x}) \qquad \textbf{(inference 1)} \tag{1}$$

$$\mathbf{c}^{*(2)} = \underset{c \ \in \mathbf{subCat} \ \&\ c \subseteq \mathbf{C}^*}{\operatorname{argmax}} \ p(c \mid \mathbf{x}) \qquad \textbf{(inference 2)} \tag{2}$$

[5] `https://keras.io/`

7 Evaluation and Results

In order to analyze the contribution and the importance of each layer in the network, we experimented with the following configurations in which different components are combined:

- **Basic CNN**: a variant with no Elmo, no features, no pooling. Fusion is kept for subtask-2.
- **No Elmo**: a variant of E-CNN without Elmo.
- **No POS**: a variant of E-CNN without POS tags.
- **No pooling**: a variant of E-CNN without pooling.
- **E-CNN (No Fusion)**: a full E-CNN for Subtask-1 and Subtask-2 (without fusion)
- **E-CNN (Fusion 1)**: a full E-CNN for both subtasks with fusion and inference technique 1 for Subtask-2.
- **E-CNN (Fusion 2)**: a full E-CNN for both subtasks with fusion and inference technique 2 for Subtask-2.

We use micro- and macro-averaged F-scores (F_1) to evaluate the performance of our two E-CNN models for both subtasks. Results are reported in Table 3.

Table 2: Macro and Micro averaged F_1 scores on the Dev and Test sets, obtained using different configurations, for both Subtasks

	Dev				**Test**			
	Subtask 1		**Subtask 2**		**Subtask 1**		**Subtask 2**	
Model	***micro***	***macro***	***micro***	***macro***	***micro***	***macro***	***micro***	***macro***
Basic CNN	88.56	78.44	79.73	70.91	87.31	78.29	77.79	71.70
No Elmo	90.76	84.10	81.47	74.15	91.39	84.89	82.92	76.02
No POS	92.99	87.42	84.30	79.08	93.89	86.42	86.31	79.57
No pooling	93.49	86.14	87.51	79.81	92.82	86.71	85.34	80.31
E-CNN (No Fusion)	**95.75***	**91.47***	79.58	76.48	**93.94***	**90.05***	80.34	76.53
E-CNN (Fusion 1)	**95.75***	**91.47***	86.23	80.21	**93.94***	**90.05***	86.53	80.49
E-CNN (Fusion 2)	**95.75***	**91.47***	**88.34***	**81.10***	**93.94***	**90.05***	**87.17***	**82.40***

For Subtask-1, our *E-CNN* model outperforms all the other models both on the dev and test sets, with an F_1 that reaches 93.94% (micro) and 90.05% (macro), on the test. It yields an improvement of +1% of F_1 micro and +3% of F_1 macro over the *No Pooling* one, and more than +6% of F_1 micro and +11% of F_1 macro over the *Basic CNN* model. This shows clearly the complementarity between the layers included in the network and the usefulness of enriching embeddings with external information. Furthermore, results show that including the POS tags and Elmo embeddings have a significant impact. This could be explained by the importance of morpho-syntactic information and the overall tweet context in detecting the correct labels. These results ranked our systems first and second in the Subtask-1 of FinNum competition. We submitted two runs (Fortia1-1 and Fortia1-2), they achieved 93.94% micro and 90.05% macro, and 93.70%

micro and 88.98% macro, respectively. Both runs use the same model (Full E-CNN) and the small variance of $\sim 1\%$ between the two runs in the F_1 macro is only due to parameters tuning.
For Subtask-2, our best model is *E-CNN Fusion 2* with averaged F_1 scores of 87.17% (micro) and 82.40% (macro). As for Subtask-1, our results show the impact of using enriched word representation and the pooling layer on enhancing context representation. We also examined the contribution of the fusion model in this subtask, and we can clearly observe the significant improvement using this mechanism (almost +7% on both F_1 micro and macro). Also, it is interesting to note that using inference 2 to predict subcategories improves the scores with almost +2%. In this subtask, our systems (Fortia1-2: 87.17% micro and 82.40% macro) and (Fortia1-1: 86.53% micro and 80.49% macro), are also ranked first and second, respectively. FORTIA1-2 uses the *E-CNN Fusion 2* model and FORTIA1-1 uses the *E-CNN Fusion 1* one.

Table 3: Macro and Micro averaged F_1 scores on the Dev and Test sets, obtained using different configurations, for both Subtasks

	Test			
	Subtask 1		**Subtask 2**	
Model	***micro***	***macro***	***micro***	***macro***
Basic CNN	87.31	78.29	77.79	71.70
No Elmo	91.39	84.89	82.92	76.02
No POS	93.89	86.42	86.31	79.57
No pooling	92.82	86.71	85.34	80.31
E-CNN (No Fusion)	**93.94***	**90.05***	80.34	76.53
E-CNN (Fusion 1)	**93.94***	**90.05***	86.53	80.49
E-CNN (Fusion 2)	**93.94***	**90.05***	**87.17***	**82.40***

8 Conclusion

In this paper, we described E-CNN architecture for numeral understanding and classification in Financial Tweets. Our models are ranked first in the FinNum shared-task both in the Subtask-1 and Subtask-2. We model the problem as a sequence labeling task where we used some knowledge base to enrich the word representation in order to enhance the context of each word. For Subtask-2, we used the *fusion* approach consisting in including information from Subtask-1 model into the Subtask-2 one to make a prediction of sub-categories, allowing to achieve very strong performance.

Acknowledgments

We would like to thank the organizers for providing the Twitter dataset and organizing the FinNum Shared Task. We also thank Youness Mansar and Guil-

laume Hubert for the interesting discussions and their valuable comments and recommendations.

References

1. Chen, C., Huang, H., Shiue, Y., Chen, H.: Numeral Understanding in Financial Tweets for Fine-grained Crowd-based Forecasting. CoRR **abs/1809.05356** (2018), http://arxiv.org/abs/1809.05356
2. Chen, C.C., Huang, H.H., Takamura, H., Chen, H.H.: Overview of the ntcir-14 finnum task: Fine-grained numeral understanding in financial social media data. In: Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies (2019)
3. Chiu, J.P., Nichols, E.: Named Entity Recognition with Bidirectional LSTM-CNNs. arXiv preprint arXiv:1511.08308 (2015)
4. Chollet, F.: GitHub, GitHub repository https://github.com/fchollet/keras (2015)
5. Fan, A., Lewis, M., Dauphin, Y.: "hierarchical neural story generation". In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 889–898. Association for Computational Linguistics (2018)
6. Ma, X., Hovy, E.: End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354 (2016)
7. Mengelkamp, A., Hobert, S., Schumann, M.: Corporate Credit Risk Analysis Utilizing Textual User Generated Content-A Twitter Based Feasibility Study. In: PACIS. p. 236 (2015)
8. Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., Smith, N.A.: Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 380–390. Association for Computational Linguistics, Atlanta, Georgia (June 2013), http://www.aclweb.org/anthology/N13-1039
9. Pagolu, V.S., Reddy, K.N., Panda, G., Majhi, B.: Sentiment Analysis of Twitter Data for Predicting Stock Market Movements. In: 2016 international conference on signal processing, communication, power and embedded system (SCOPES). pp. 1345–1350. IEEE (2016)
10. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
11. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep Contextualized Word Representations. CoRR **abs/1802.05365** (2018)
12. Souza, T.T.P., Kolchyna, O., Treleaven, P.C., Aste, T.: Twitter Sentiment Analysis Applied to Finance: A Case Study in the Retail Industry. arXiv preprint arXiv:1507.00784 (2015)
13. Wu, M., Liu, F., Cohn, T.: Evaluating the Utility of Hand-crafted Features in Sequence Labelling. arXiv preprint arXiv:1808.09075 (2018)
14. Xu, Y., Cohen, S.B.: Stock Movement Prediction from Tweets and Historical Prices. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1970–1979 (2018)

15. Yang, S.Y., Mo, S.Y.K.: Social Media and News Sentiment Analysis for Advanced Investment Strategies. In: Sentiment Analysis and Ontology Engineering, pp. 237–272. Springer (2016)