BRNIR at the NTCIR-14 finnum task: Scalable feature extraction technique for number classification*

Alan Spark[1][0000−0002−5112−4842]

AUTO1 Group, Bergmannstr. 72, 10961 Berlin, Germany
alan.spark@auto1.com
https://www.auto1-group.com/

Abstract. The following paper describes effort and results in NTCIR14 FinNum Task, approaches and results in both subtasks of FinNum. In the task, the team focuses on feature extraction and experiments on the concatenation of various features, on insights derived from an unsupervised approach to construct and extend data available for analysis. Feature extraction steps are designed for parallel execution thus proposed technics meant for use at scale.

Keywords: feature extraction · numeral understanding · FinTech

Team Name: BRNIR
Subtasks: FinNum Subtask 1, FinNum Subtask 2

1 Introduction

In the era of FinTech, it is hard to overestimate the importance of automated financial analysis. Automated understanding of numeral content in a digital document at scale has enormous potential in the financial domain.

To enhance techniques for number understanding The Fourteenth NTCIR Conference on Evaluation of Information Access Technologies (NTCIR-14) includes new FinNum Task: Fine-Grained Numeral Understanding in Financial Social Media Data[5].

In the task, participants were asked to map a given number to one of seven categories in Subtask1 and one of seventeen categories in Subtask2. The taxonomy details are shown in Table1. Experts manually annotate the ground truth (training, dev, test datasets) with Kappa agreement between each pair of experts 70.30%, 69.75% and 67.07%[4]

In NTCIR-14 team BRNIR participates in the task, including both subtasks. The primary goal of the team is the extraction of features sets applicable for both subtasks. One might expect work on unsupervised scalable approach for feature extraction described further in the paper. This paper describes carried work by

* Supported by AUTO1 Group

Table 1. FinNum taxonomy

Category	Subcategory	Category	Subcategory
Monetary	money quote change buy price sell price forecast stop loss support or resistance	Percentage	relative absolute
		Option	exercise price maturity date
		Indicator	indicator
		Temporal	date time
		Quantity	quantity
		Product	product

the following structure: Section 2 describes feature extraction in great details; Section 3 conducts experiments on different combination of features with fixed model architecture; Section 4 elaborates on submitted runs and error analysis; Section 5 concludes the paper.

2 Feature representation

2.1 Overview

The understanding of numbers in text corpus is a potential area to apply intelligence tools and techniques; however, we need to think in scale to use it effectively. Amount of financial tweets, news, reports is growing exponentially, and one needs to work on scalable methods according to the growth of the data. Previous work on the FinNum[4] includes some annotation effort to aggregate tokens like *%, percent, pc* as one token, the same work done to aggregate options, indicators, temporal tokens. This approach shows amazing results however author decided not to follow it and work on a scalable approach to achieve the same.

With the challenge of data scale, the team sees the necessity of massive parallelization on every step of AI pipeline construction. To achieve it, BRNIR team proposes a multistep pipeline of feature extraction. The source code is available at URL[16].

In this chapter, focus is expressly on analysis and findings of the feature extraction, at first section describes preprocessing step; and then steps to construct features of *tweet features* types and finally construction of *number features* type. Author believes that tweet features should contribute to the classification

2.2 Feature extraction

To achieve pledged scalability, the team designs every step to be executed independently from others feature extraction steps, in the best scenario every tweet in a processing step should be treated without any information about other tweets hence we can achieve massive parallelization. However, there are exceptions: for

example preprocessing has to be executed first and topic modeling cannot avoid a global state. The author finds that all features which one might extract from

Table 2. Features type summary

Tweet features	
Topic distribution	a vector with topics distribution of a tweet
Tickers	multi-label encoding of tickers presented in a tweet
Tags	multi-label encoding of tags presented in a tweet
Number features	
Number properties	a vector encoding a number properties such a value, position, type and other.
Token context	"Bag-of-words" like encoding of tokens neighboring a number.
Character context	"Bag-of-words" like encoding of characters neighboring a number

the corpus of tweets are naturally divided into two general groups: *tweet features* and *number features*. Tweet features are these which represent a tweet as it is, the number features describes the number itself and context in which it appears. The summary of feature types can be found in Table2.

The team sees the tweet features group as important features group, analysis of data shows that majority of tweets has one number to classify or number/s belong to one class/subclass thus full context of a tweet might contribute to these cases. Figure1 shows corresponding distributions.

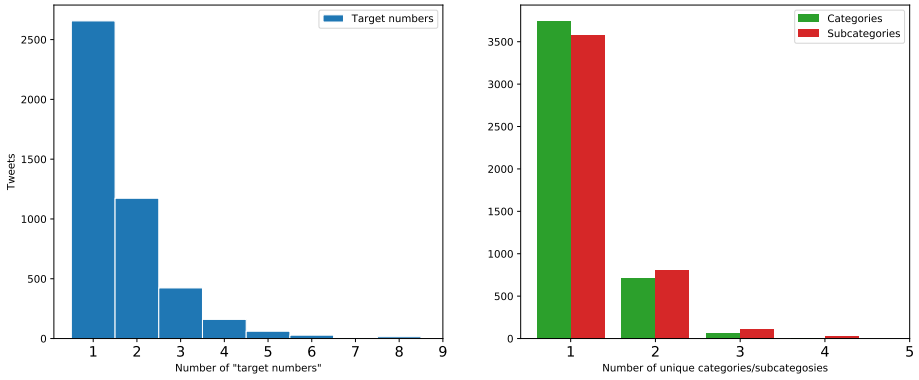

Fig. 1. Number of "target numbers" per tweet on the left, Number of unique categories/subcategories per tweet on the right

Preprocessing In this subsection, paper describes the preprocessing step - this step has to be run first. However, every tweet might be processed independently

so that one still can rely on parallel execution. The script to run the first step is located in *feature_extraction/00_00_tokenization.py*[16].

System tokens Team introduces the concept of system tokens as entities which may be treated independently from their value, e.g., an URL, when replaced by a system token encodes a presence of an URL regardless where it links too also an URL's system token won't be split in a list of individual characters at the later steps thus won't add noise to the system. Author defines a set of system tokens for *tickers, tags, URLs, floats, integers*, for more details on what these entities are, please refer to corresponding feature steps. Toward each entity detection the team construct a regular expression, please see Table3 and source code[16] for exact regular expression and corespondent system token. The values for system tokens were preserved as lists in the order of appearance, thus one may access these values at later feature extraction steps.

Table 3. System tokens summary

Entity	**System token**	**Regular expression**
Ticker	systicker	`r"\$[A-Z-_\.]{1,8}"`
Tag	systag	`r"#[A-Za-z]{1,15}"`
URL	sysurl	`r"https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+(?:/[-_\w]+)*/?(?:\?[=&\w\d\-\_]+)*"`
Float number	sysfloat	`r"\d*(?:\,\d+)*\.\d+"`
Integer number	sysint	`r"\d+(?:\,\d\d\d)*"`

Tokenization Following the initial attempt to create an unsupervised feature extraction application author doesn't rely on metadata coming along a tweet, specifically, data given in **target_num**, instead the team extracts all tokens directly from a tweet's raw text. To extract all tokens: the team first replaces all entities described in the subsection above by corresponding system tokens, then converts all characters to lower case, and utilizes **Tokenization Algorithm 1**, python implementation available at repository[16]. The tokenization approach has its limitation and might skip entities in the format what won't match a corresponding regular expression, however, in this particular run the approach finds more numbers from the raw text than presented in **target_num** properties of training data. Moreover, the algorithm doesn't miss any from the numbers given in **target_num**. Which alone opens an opportunity to Semi-supervised learning technics and extension of the training data set. The approach detects extra numbers in more than 32% of tweets in given corpus, e. g. Tweet (T.1) contains 3 numbers, but only 2 are given in **target_num**: [”10”, ”12.”] all of them were discovered during tokenization together with one extra number, **discovered numbers**: [10, 12, 16]

Algorithm 1 Tokenization algorithm

```
1: procedure GetCharType(c)
2:     if c in "digits" then return "digits"
3:     if c in "letters" then return "letters"
4:     if c in "specialchars" then return "specialchars"
5:     if c in "spaces" then return "spaces"
6:
7: procedure Tokenization(tweet)
8:     prevCharType ← "None"
9:     token ← List
10:    tokens ← List
11:    for all char in tweet do
12:        charType ← GetCharType(char)
13:        if charType! = prevCharType then
14:            if length(token) > "0" then
15:                tokens ← tokens.push(token)
16:            if charType == "spaces" then
17:                token ← List
18:            else
19:                token ← [char]
20:        else if charType == specialchars then
21:            if length(token) > 0 then
22:                tokens ← tokens.push(token)
23:            token ← [char]
24:        else
25:            token ← token.push(char)
26:        if charType == "spaces" then
27:            prevCharType ← "None")
28:        else
29:            prevCharType ← charType)
```

(**T.1**) *$FNKO $10 is a no-brainer. Should trade back to IPO price $12. Remember, initial range on IPO was $16 on high end. Quiet period expiry soon.*

Topic modeling One of the features which describe a tweet is a topic distribution of that tweet, to extract hidden information and connections between tweets, the team applies Latent Dirichlet allocation[2] to all tweets and use it as some form of tweets embedding. Since Chang et al.[3] challenge common usage of likelihood for evaluation of topic modeling, author follows previous work on topic coherence[14][10][1][13] to automatically pick the "best" number of topics, the team utilizes Scikit-learn[12] implementation of a coherence score.

The author believes this is an interesting Implicit/Latent Feature, and the team finds that it alone can adequately represent a tweet. More details are given in the Experiment section. The script for the step is located in *feature_extraction/01_01_topics.py*[16].

Incorporate tags Almost every financial tweet inherently has some tags used across of Twitter service, e.g., in the tweet (T.2) *#NewPartnersnip* is a tag. The team includes this tagging as a feature which additionally represents a tweet in the newly defined tag's space. For the current study, author decides to consume tags as a discrete feature and enumerate all of the tags, combine them using multi-labels encoding to construct a binary feature vector. The script for the step is located in *feature_extraction/01_02_tags.py*[16].

(**T.2**) *$GPRO This is interesting... concierge says they sell about 100 a week. GoPros marketing is killer! #NewPartnership*

Incorporate tickers Often a tweet includes a *ticker* - the codified name of a company traded at one of a stock market, e.g. in the tweet (T.3) *$NAK* refers to Northern Dynasty Minerals Ltd. traded at *The New York Stock Exchange*. In the similar way with tags, team uses multi-labels binarization as base level, with the room for further improvements. The script for the step is located in *feature_extraction/01_03_tickers.py*[16].

(**T.3**) *$NAK price breaks above daily ichi cloud , above downtrend line that connects pivots of 11/14 and 11/24 and above 8ema and 50sma: very*

A number as a feature vector The team represents every target number in a tweet as a feature vector encodes the following information:

- *numeric* raw-value,
- *boolean 1 or 0*: is number an integer,
- *boolean 1 or 0*: is number a floating point number,
- *numeric* position in a tweet as a ratio between token position/length in tokens,
- *numeric* log of raw-value

The script for the step is located in *feature_extraction/02_01_numtypes.py*[16].

Continuous Feature normalization As we think of feature extraction without consideration of any model we would like to normalize continuous feature. As shown before [6],[8] normalization/scaling of continuous features might be extremely important, thus author normalize numeric raw-values and use it instead. The team implements an extra pass to apply Softmax normalization (1) for every raw numeric value, in addition to $\tilde{x}$ $\tilde{x}^2$ and $\sqrt{\tilde{x}}$ are included to the feature vector.

$$\tilde{x_i} = \frac{e^{x_i}}{\sum_j^N e^{x_j}} \tag{1}$$

However, this normalization was applied after the submission. In the submission, only log scaling is utilized.

A number context To define a context in which a number appears, the team utilizes two approaches to look at data around a target number. The first one at the scale of tokens, defined above, and the second at the scale of individual characters. Later during experimentation, the team extracts context with different amount of consecutive neighbors.

Tokens context The team uses tokens, created at the very first prepossessing step, and uses Bag-of-words model to construct a feature vector. To pick terms which represent a number context author follows the approach used by Mikolov at al.[11] in the Skip-gram model. Figure 2 shows an example of token context with **n** set to two. The script for the step is located in *feature_extraction/02_02_context_terms.py*[16].

Fig. 2. Token context for target number, where n=2. The brown areas highlight selected tokens

Characters context In addition to the Tokens context, BRNIR team defines a character context to be a **n** consecutive characters before a target number and a **n** consecutive characters following a target number. The team creates a space for unigram characters by mapping every token to a character vector and then flat map it back to original vector, Figure 3 shows an example of characters context with **n** set to four. For every tweet its Character context is created by using Bag-of-words model to construct a feature vector similarly with Token context. The script for the step is located in *feature_extraction/02_03_context_symbols.py*[16].

d,SysInt,i,m,p,o,r,t,a,n,t SysInt %,t,r,i,g,g,e,r,s,u,o,

Fig. 3. Character context for target number, where n=4. The orange areas highlight selected characters

3 Experiments

As stated above the primary goal of the work is feature extraction, so that the current section is devoted to experiments on the concatenation of different features. The team reports performance in metrics chosen by NTCIR-14 organizers - micro and macro average F1 score.

3.1 Architecture

The team follows the approach used by video recomender[6] and concatenate heterogeneous features as the first layer of a network (input), succeeded by layers of fully connected Rectified Linear Units (ReLU)[7] with a dropout[15] in between. In other means, the team follows[4] and use cross-entropy Loss function, the Adam optimizer[9] for the model parameters. Fitting the model is limited by 30 trial epochs. Figure4 shows the general network architecture and features concatenation input. Network structure follows a typical scheme - every succeeding

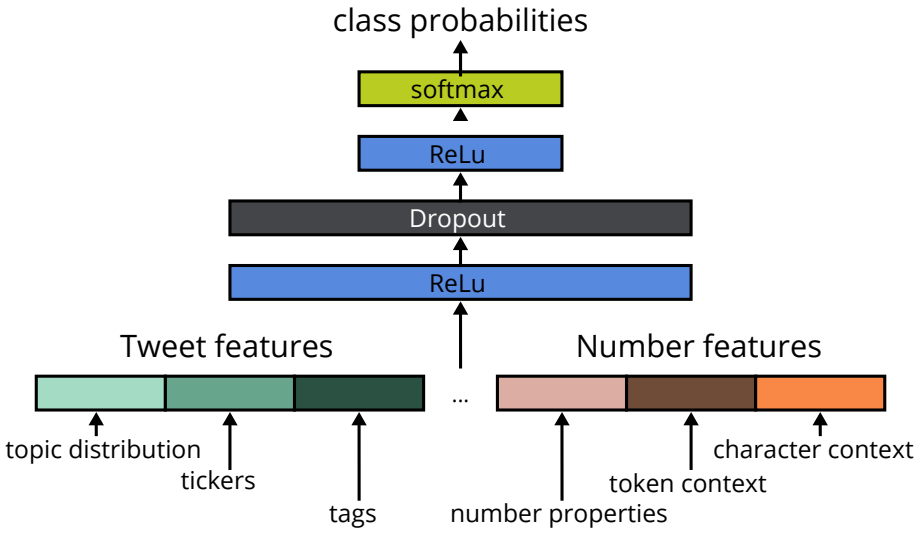

Fig. 4. Classification model architecture

layer diminishes in size by half, the last "output" layer sized accordingly to the number of classes in a task:

– **Subtask 1**: 512 ReLU → 256 ReLU → 7 SoftMax
– **Subtask 2**: 512 ReLU → 256 ReLU → 17 SoftMax

Once again, the author highlights the fact that during experiments the model architecture is fixed. The classification model is the "independent variable" of the experiment; hence all the work is done on the features extraction and observation of performance changes regarding changes on features combination.

3.2 Feature experimentation

Choosing a features set to construct an input to a model is a challenge, the decision impacts the performance drastically. To test the author's assumptions on feature concatenation, BRNIR builds a performance framework which implements 10-fold cross-validation. The framework shuffles all entries in the training set and splits it into ten even samples; then iterates over samples selecting current as a test sample and nine others as a training sample. As a final evaluation result, the frameworks return an average of ten runs for micro and macro average F1 scores. Figure5 contains most important results.

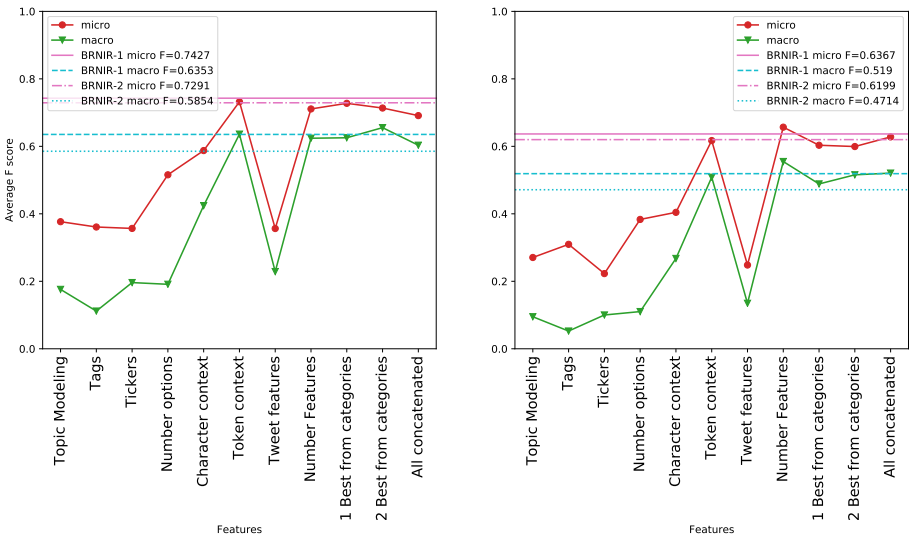

Fig. 5. Performance of individual features sets on the training data, horizontal lines depict submitted runs. Subtask 1 on the left, Subtask2 on the right

At first, the team runs experiments for individual features sets. Evaluation results for distinctive features in "tweet" category show the least performance

among all others, for absolute metrics the combination of all "tweet" features doesn't show significant if any performance gain either. However, the gap between micro and macro average F1 score is reduced drastically for about 50% for *Subtask 1* and about 56% for *Subtask 2*. This improvement shows that regarding unbalanced categories appearance the combination of the all "tweet" features provides helpful information and better generalization ability to the model. Here author highlight that "Topic distribution" shows the best performance as a single "tweet" feature in seven categories classification hence supports the assumption that hidden features may healthily represent an entity.

Evaluation results in "number" feature group also show better generalization for all three features set concatenated together against "individual" runs. The "Token context" feature shows best micro average F1 score of 0.732079 however miss category "Product Number" in one run thus the team doesn't pick it as a feature set for any of submissions.

Further experiments include concatenation of the following feature the best performers from each category: Topic distribution and Number context. Next, two best from each feature group, and finally the sequence of all the features. The team picks two last combinations to train the model and submit the results as BRNIR-1 and BRNIR-2 respectively. The official evaluation shows similar results with reported above. BRNIR-1 performs even better which indicates that the classification model achieves good generalization power without overfitting problem.

4 Submitted runs and Error analysis

The team utilizes the approach described in the sections above to produce the following runs for the Fine-Grained Numeral Understanding in Financial Tweets, both subtasks:

- **BRNIR-1**: Features concatenation (Topic distribution + Tags + Number Options + Token context),
- **BRNIR-2**: Feature concatenation (All features)

To evaluate our work we submit both runs. Please see Table 4 for results in terms of The micro and macro average F1 scores. In the *Subtask 1*, the run BRNIR-1 outperforms BRNIR-2, the team expects exactly this based on finding highlighted in section Experiments. However, the author didn't expect BRNIR-1 outperforms BRNIR-2 in the *Subtask 2*, based on the same findings, see Figure5. Nevertheless, the author believes that further experiments with the model architecture and synthetic over/under-sampling are required to pick the best feature concatenation. In the following section, the team highlights findings on True labels and error analysis.

4.1 Confusion Matrix Analysis

The Figure6 and Figure7 show the confusion matrices for both submitted run, with Subtask1 on the top and Subtask2 on the bottom. The team finds that

Table 4. Evaluation results for FinNum task

	Micro-averaged F-scores	**Macro-averaged F-scores**
Subtask 1		
BRNIR-1	0.7427	0.6353
BRNIR-2	0.7291	0.5854
Subtask 2		
BRNIR-1	0.6367	0.5190
BRNIR-2	0.6199	0.4714

in Subtask1 both proposed solution hit accuracy of about 80% in *Monetary, Percentage* and *Temporal* categories. Adding extra features in BRNIR-2 helps to burst accuracy for category *Option* from 45% to 73%. However, the same addition leads to the decline in accuracy for categories *Product Number* and *Quantity*: from 23% to 5% and from 42% to 29% respectively. The classification shows the worst performance in the categories Product Number and Quantity. Most often Product Number is confused with Temporal category: 36% in run1 and 41% in run2.

The author suspects that the tweet features being responsible for that. Two observations are leading the author to these views. At first, statistics on neighbor numbers in a tweet, Table5, shows that the most common neighbor for Product Number is a number represents a Temporal category. At second, by adding more tweet features in the BRNIR-2 run, we see the decrease in accuracy for Product Number and increase misclassification of numbers in category Product Number as a Temporal category.

Table 5. Numbers of neighbour categories

	Indicator	**Monetary**	**Option**	**Perc.**	**Pr. Number**	**Quantity**	**Temp.**
Indicator	193	44	X	11	X	7	29
Monetary	44	3467	60	173	22	242	776
Option	X	60	308	10	X	24	4
Percentage	11	173	10	943	7	61	232
Product Number	X	22	X	7	128	15	38
Quantity	7	242	24	61	15	694	185
Temporal	29	776	4	232	38	185	3613

The Quantity category is the second worst performing category, which also suffers from adding more features in the BRNIR-2 run, with the slight difference

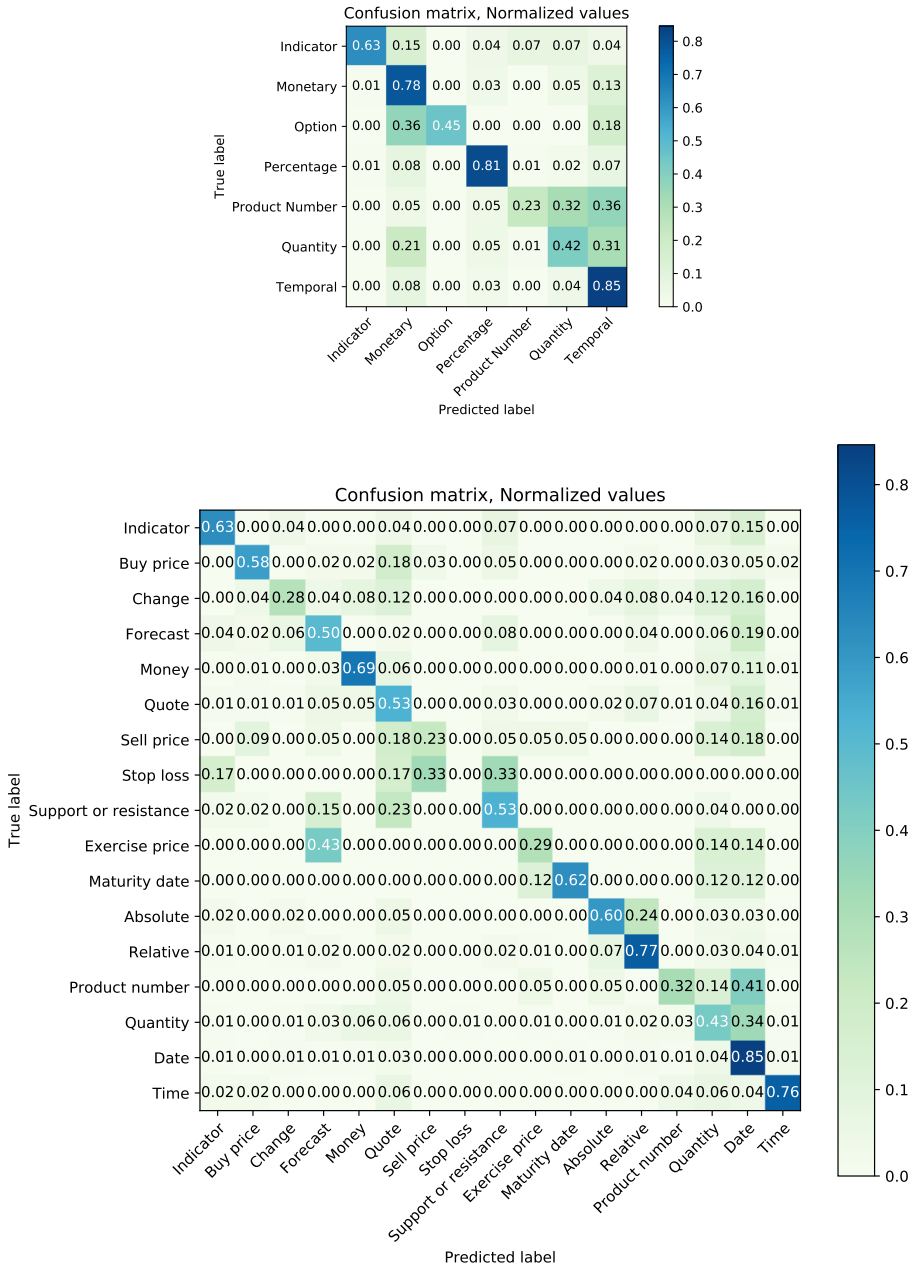

Fig. 6. Confusion matrices for run BRNIR-1: Subtask1 (7 categories) on the top, Subtask2 (17 categories) on the bottom

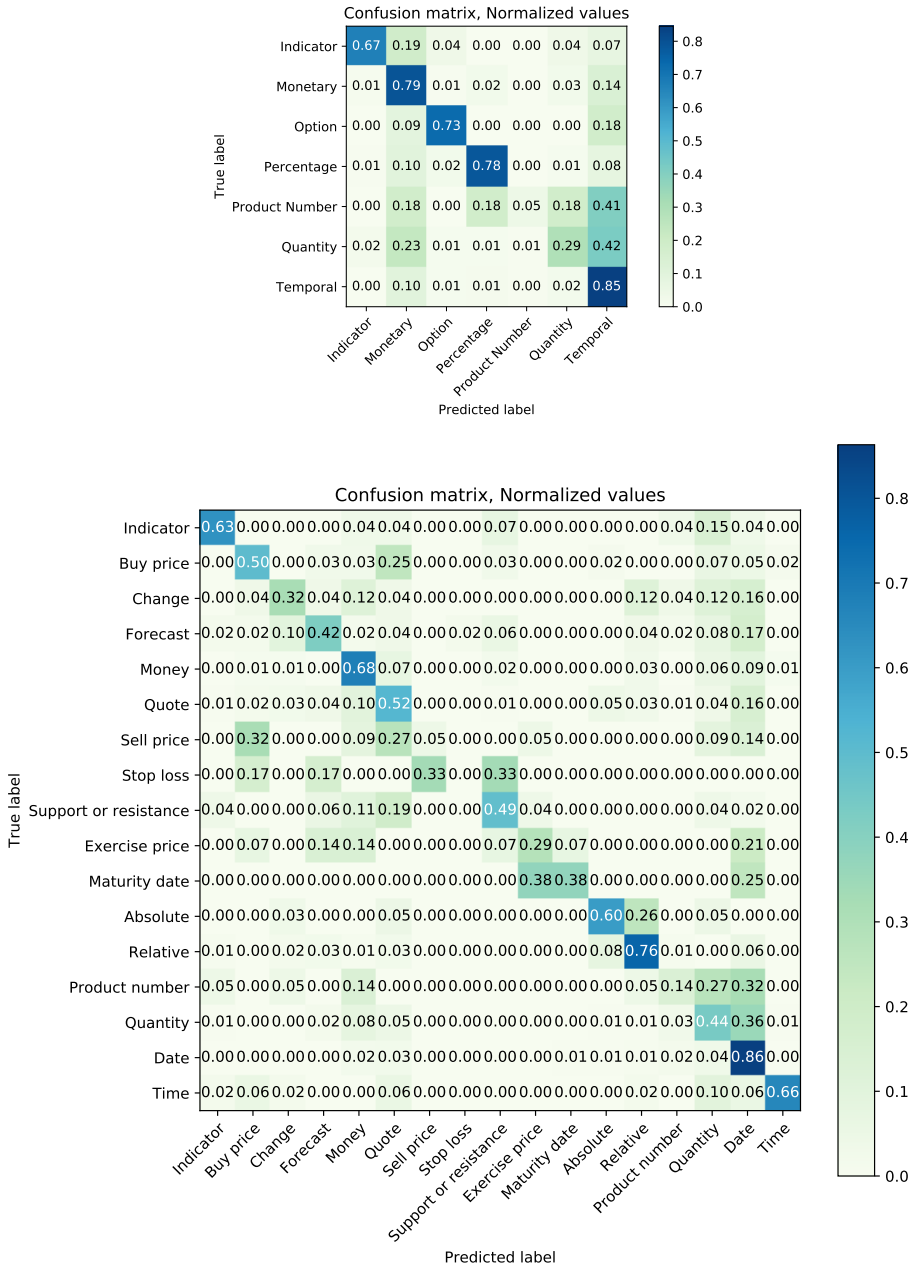

Fig. 7. Confusion matrices for run BRNIR-2: Subtask1 (7 categories) on the top, Subtask2 (17 categories) on the bottom

the most common neighbors are Monetary and Temporal, coexist in 242 and 185 tweets respectively

In Subtask2 one might see very similar patterns: Date and Time, only two Temporal subcategories, show the accuracy of around 80% and 70% in both runs. Subcategory of Percentage - Absolute and Relative, in both runs, perform at about 60% and 75% of the accuracy. The Indicator hits 63% again in both runs. Once again the numbers in the second most present category, Temporal 34.03%[5], show the best accuracy in the classification. However, the subcategories, in the most presented category, Monetary 35.94%[5], after split into eight categories show mixed performance. Stop loss subcategory achieve accuracy of 0 at the same time this subcategory the less presented subcategory in the FinNum 2.0 data set just 0.39%. The author is seeking a more in-depth analysis of results in Monetary sub/category analysis in future work.

5 Conclusion

This paper introduces unsupervised approaches for feature extraction in application to NTCIR-14 FinNum task. Also, the proposed methods are parallelizable and meant to be run at scale. Thus, the analysis of numerical data is feasible at web/industrial scale. The evaluation results show that the introduced approach is helpful. For classification in both subtasks, the team uses the same feature set extracted from the given data. The team sees future work in addressing natural imbalance of the data and extending training data with unsupervised technics on additional numbers extracted from training data.

References

1. N. Aletras and M. Stevenson. Evaluating topic coherence using distributional semantics. In Proc. of the 10th Int. Conf. on Computational Semantics (IWCS13), pp. 13-22, 2013.
2. David M. Blei , Andrew Y. Ng , Michael I. Jordan, Latent dirichlet allocation, The Journal of Machine Learning Research, 3, pp. 993-1022, 3/1/2003
3. J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei. 2009. Reading tea leaves: How humans interpret topic models. In Advances in Neural Information Processing Systems 21 (NIPS-09), pages 288-296, Vancouver, Canada.
4. C.-C. Chen, H.-H. Huang, Y.-T. Shiue, and H.-H.Chen, Numeral understanding in financial tweets for fine-grained crowd-based forecasting, in 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 136-143, IEEE, 2018.
5. C.-C. Chen, H.-H. Huang, H. Takamura, and H.-H. Chen, 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.
6. P. Covington and J. Adams and E. Sargin: Deep Neural Networks for YouTube Recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, NY, USA 2016

7. X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. J. Gordon and D. B. Dunson, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11), volume 15, pages 315323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.
8. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167, 2015.
9. D. P. Kingma, and J. Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
10. Jey H. Lau, D. Newman, T.Baldwin. Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality. 14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014. 530-539. 10.3115/v1/E14-1056.
11. T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient estimation of word representations in vector space - arXiv preprint arXiv:1301.3781, 2013
12. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. Scikit-learn: Machine Learning in {P}ython}, Journal of Machine Learning Research (12) pp. 2825 - 2830, 2011
13. F. Rosner, A. Hinneburg, M. Ro der, M. Nettling, and A. Both. Evaluating topic coherence measures. CoRR, abs/1403.6397, 2014.
14. M. Rder , A. Both , A. Hinneburg, Exploring the Space of Topic Coherence Measures, Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, February 02-06, 2015, Shanghai, China
15. N. Srivastava, G. Hinton, A Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15 (1), 1929-1958, 2014
16. Code repository https://github.com/Arisiru/ntcir-14-finnum. Private repository please contact author for the access. The repository doesn't contain data, please refer [4] for data access.