ASNLU at the NTCIR-14 finnum task: Incorporating Knowledge into DNN for Financial Numeral Classification

Chao-Chun Liang and Keh-Yih Su

Institute of Information Science, Academia Sinica, Taiwan
{ccliang, kysu}@iis.sinica.edu.tw

Abstract. This paper describes our work for solving the financial numeral classification problem in the NTCIR-14 FinNum task, and discusses experimental results. After implementing the three proposed vanilla neural network models (CNN, RNN, and RNN with CNN filters), we further incorporate POS and NE linguistic features. Inspired by human observation, we also propose a preprocessing procedure, which splits numerals in the Twitter string in advance, to reduce the OOV rate in the test set. Experimental results show both approaches improve the performance significantly.

Team Name. ASNLU

Subtasks. Fine-Grained Numeral Understanding in Financial Tweets (7-Category classification (English) and 17-(Sub)Category classification (English))

Keywords: Financial Numeral Classification, Numeral Understanding, Financial Social Media, Natural Language Understanding

1 Introduction

Fine-Grained Numeral Understanding in Financial Tweets (*FinNum*) is a task [1, 2] in NTCIR-14, which classifies a given numeral in a financial tweet to a numerical category. Its two associated subtasks include 7-Category classification, which classifies a numeral into seven categories, and 17-(Sub)Category classification, which extends the classification task to the subcategory level and classifies a numeral into 17 subcategories. In this task, the ASNLU team participated in both subtasks, submitting the respective results from two separate runs for each subtask. To avoid heavy handcrafted feature engineering, we used neural networks for the tasks.

However, deep learning typically requires large amounts of data for parameter training, and most related approaches make no use of existing linguistic knowledge. In contrast, equipped with linguistic knowledge, human learning is much more efficient [3, 4]. This suggests that incorporating linguistic knowledge into a neural model

2

Table 1. FinNum categories

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

Table 2. Sequence labeling for FinNum classification

Tweet	**8**	breakouts:	$CHMT	(stop:	**$17.99**	).
Main Category	**Quantity**	O	O	O	**Monetary**	O
Sub Category	**Quantity**	O	O	O	**stop loss**	O

could improve performance. We thus propose various ways to integrate part-of-speech (POS) and named-entity (NE) linguistic features into the DNN model. Furthermore, based on human observation, we also propose suitable pre-processing (i.e., splitting numerals in the Twitter string in advance) to reduce OOV rates. Experimental results show that both proposed approaches improve performance significantly.

2 Proposed Approaches

Given an input word sequence $W=\{w_1, w_2, \ldots, w_n\}$, our goal is to classify each associated target numerical word as its corresponding numerical category. We model this task as a sequence labeling process [5]. Each word w_i is tagged by a label, which is a member of the union of 'O' and the pre-specified FinNum-Category set (listed in **Table 1**), where 'O' indicates that the associated word is not a targeted word to be classified. **Table 2** shows an example of the corresponding numerical category labeling for a tweet.

We implement three end-to-end DNN models: a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN), and a combination of RNN and CNN (RNN+CNN), and integrate POS and NE linguistic features into these models. Details are given below.

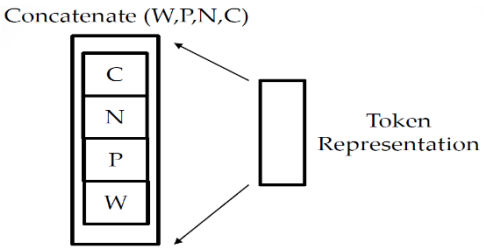

Fig. 1. Proposed token representation. W: word embedding; P: part-of-speech, N: named-entity type; C: category-pattern feature.

2.1 Data Preprocessing

We first split the given Twitter content into various sentences using three special punctuation symbols ('=', '?', and ','), and then split each sentence into its various tokens which are separated by space characters. The tokenized sentences are then passed to the CoreNLP [6] package to extract the corresponding POS and NE information.

2.2 Token Representation with External Knowledge

The proposed **token representation** is a concatenation of four different kinds of information (displayed in **Fig. 1**). Each is specified as follows.

- **Pre-trained word vectors** (denoted as W). We use Glove.840B.300D word embeddings [7], which were trained from a corpus of 840 million tokens with 300d vectors.
- **Part-of-speech** (P) **and named-entity** (N) features. They are extracted using CoreNLP as one-hot encodings.
- **Category Pattern** (C) features. Local patterns in the FinNum tweet corpus usually imply special meanings. For example, numerals combined with capital characters can represent a product number (e.g., the numeral "334" in "APD334"). We collect six common patterns (associated with company, money, product number, date, time, and number) in the FinNum tweet corpus, and represent them with one-hot encodings. We extract these category patterns with the following simple heuristic rules.
  - **company:** capital letters following a '$' (e.g., '$NTNX').
  - **money:** a numeral following a '$', or a '$' followed by a numeral (e.g., '$20' or '13$').
  - **product number**: capital letters followed by a numeral (e.g., 'CYC065').
  - **date**: either 'MM/DD/YY' or 'MM-DD-YY' (e.g., '11/09/17' or '11-09-17').

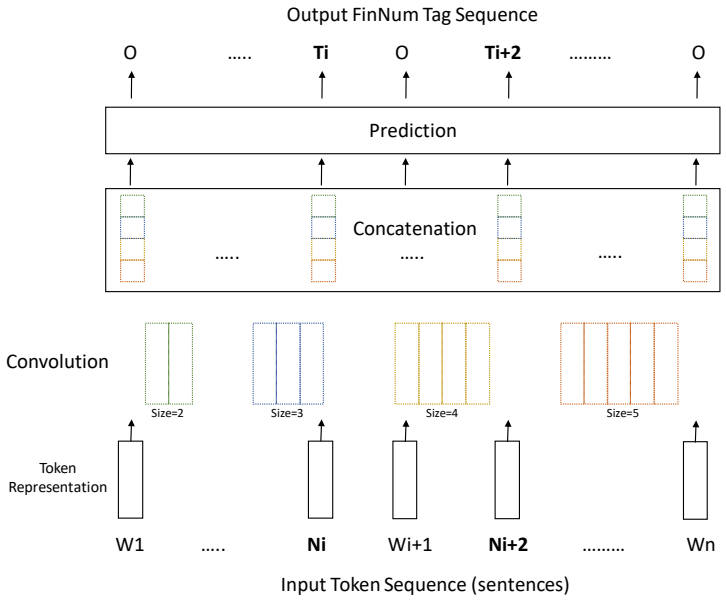

Fig. 2. Proposed CNN architecture

- **time:** either 'HH:MM' or a numeral followed by a time keyword[1] (e.g., '6:45' or '3:25 p.m.').
- **number:** numerical words (e.g., '68').

2.3 CNN-Based Approach

We observe that the numeral categories are frequently associated with local patterns in the tweet corpus. For example, a numeral followed by '%' (e.g., "85%") usually corresponds to the 'percentage' category. Since CNN has been successfully applied to many NLP tasks [8, 9] to recognize local patterns, we use it to detect local patterns for this task. Furthermore, we use CNN filters with four different window sizes to capture patterns with different lengths.

The proposed CNN architecture consists of an input token representation layer, four different kinds of CNN filters (with window sizes of 2, 3, 4, and 5), one merged layer (the concatenated outputs from the four kinds of convolutional filters), and one fully connected prediction layer. In addition, we set the dropout rate to 0.5. We generate output labels using the softmax function. The proposed architecture is shown in **Fig. 2**.

[1] The time keyword list includes {'am', 'pm', 'AM', 'PM', 'a.m.', 'A.M.', 'p.m.' and 'P.M.'}.

2.4 RNN-Based Approach

Due to its ability to capture context information, the RNN model is widely used in sequence labeling tasks such as machine translation [10] and named entity recognition [5]. We use a vanilla RNN model and encode the input token sequence using its corresponding token representation and then feed this into a single layer of bi-directional Gate Recurrent Units (GRUs) and output the prediction score of each possible category for each target token. The proposed RNN architecture consists of one input token representation layer and one bidirectional GRUs layer with 128 hidden nodes. The dropout rate is set to 0.5, and we generate the output label using a softmax function. The proposed architecture is shown in **Fig. 3**.

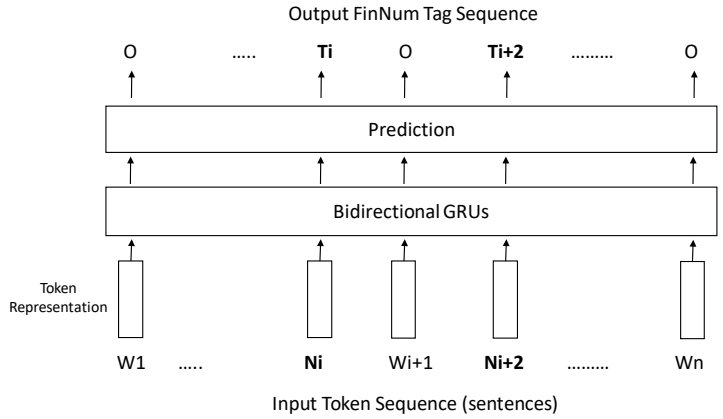

Fig. 3. Proposed RNN architecture

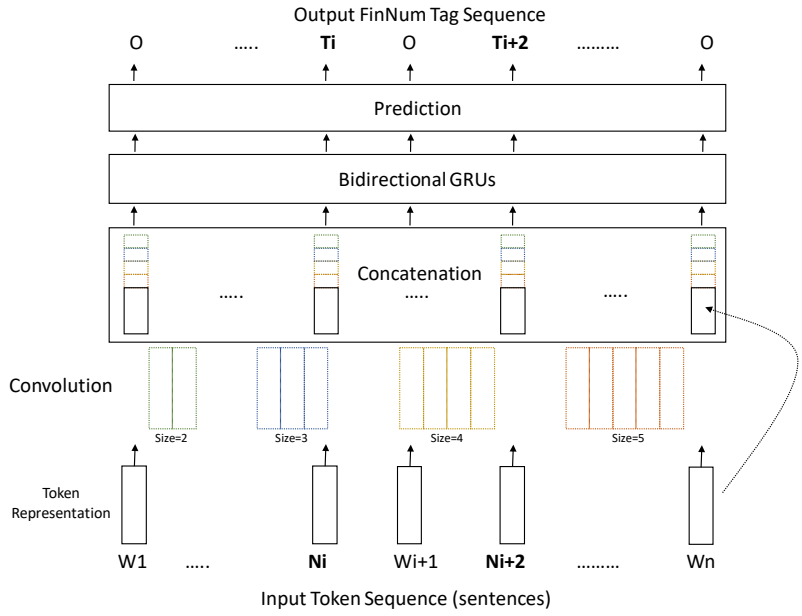

Fig. 4. Proposed RNN architecture with CNN filters

6

2.5 Combining RNN with CNN Features

To capture local pattern information in the RNN model, we further concatenate the CNN outputs (generated from the CNN filters) with the token representation. The structures of the CNN filters and the RNN model mirror that described in Sections 2.3 and 2.4, respectively. The concatenated token representation is fed into a single layer of bidirectional GRUs. Likewise, the dropout rate here is also set to 0.5, and we generate the output label using a softmax function. The proposed architecture is shown in **Fig. 4**.

2.6 Rescoring

For the best results, we exclude the 'O' (i.e., the out-of-category class) label from the category candidates for each target numeral. This was to avoid generating invalid non-numerical category labels for each specified numeral.

3 Experiments

3.1 Dataset and Evaluation Metrics

We conducted experiments on the NTCIR-14 FinNum dataset[2], which contains 4,072 tweets with numerals in the training set, 457 tweets in the development set, and 786 tweets in the test set. Per the official specification, we use the micro/macro-averaged F-scores to evaluate the classification results.

3.2 Official Evaluation Results

We submitted two-run results for (1) 7-Category classification and (2) 17-(Sub)Category classification tasks in English. Our official evaluation results are listed in **Table 3** and **Table 4**. Below we describe our experimental settings for each run.

Table 3. Performance of 7-Category classification

Run	Micro	Macro
ASNLU-1	89.40%	79.96%
ASNLU-2	89.72%	80.93%

Table 4. Performance of 17-(Sub)Category classification

Run	Micro	Macro
ASNLU-1	79.12%	72.51%
ASNLU-2	77.37%	70.09%

[2] https://sites.google.com/nlg.csie.ntu.edu.tw/finnum/data

- 7-Category classification
  For the two runs, in general each input token was represented by Glove.840B.300D pre-trained embeddings and POS and NE features. In addition, we used the model of combining RNN with CNN features; we used the Adam optimizer, a 0.5 dropout rate, early stopping (stopping-patience=10), and a stopping-min-delta of 1e-4.
  ─ Run-1 (ASNLU-1): CNN Filters with *Rectified Linear Unit* activation function.
  ─ Run-2 (ASNLU-2): CNN Filters with *Hyperbolic tangent* activation function.

- 17-(Sub)Category classification
  The common settings for two runs are identical to the above Run-1 7-Category classification task.
  ─ Run-1 (ASNLU-1): with additional *Category Pattern* features on input word representations.
  ─ Run-2 (ASNLU-2): without additional *Category Pattern* features on input word representations.

3.3 Discussion

To show the ability of each model given POS and NE linguistic features, we present their test set performance in **Table 5**.

Table 5. Test set performances with different kinds of knowledge. "**None**" denotes the NN models without incorporating any knowledge. "**POS&NE**" denotes the NN models with both POS and NE information. "**Pattern-Rule**" denotes the NN models that incorporate category patterns specified by handcrafted rules.

Knowledge	CNN		RNN		RNN with CNN filters	
	Micro	Macro	Micro	Macro	Micro	Macro
None	81.83%	69.54%	**84.22%**	73.36%	82.71%	69.63%
+ POS&NE	88.21%	79.14%	88.45%	78.63%	**89.72%**	80.93%
+ POS&NE + Pattern-Rule	87.73%	78.47%	88.76%	83.55%	**89.24%**	81.50%

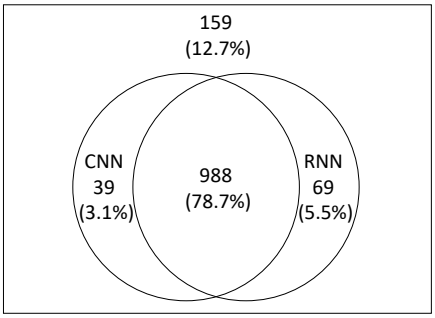

Fig. 5. Division of classification results by CNN and RNN models

Performance without linguistic knowledge: Not surprising, this group yields the worst results in comparison with that incorporating linguistic knowledge. Within this group, the RNN-based model has the best performance (84.22% micro F-score, the first row of **Table 5**). **Fig. 5** shows the division of numerical classification performance between the CNN and RNN models: 78.7% of the instances are classified correctly by both models, 3.1% are classified correctly by the CNN model only, 5.5% are classified correctly by the RNN model only, and 12.7% are correctly classified by neither CNN nor RNN.

Further analysis reveals that most errors made by RNN were due to the model missing local patterns associated with the numeral such as "num/num" (Temporal) in "10/24" or "num%" (Percentage) in "7.8**%**". In contrast, CNN misses context information associated with the numeral such as "*sold*" in "*you **sold** ESPR at **11** and CLVS at **29** but thanks for this tip*". Here the content word '*sold*' gives the clue that numerals "11" (Monetary) and "29" (Monetary) are to be assigned to the "Monetary" category.

Performance incorporating linguistic knowledge: The second row in **Table 5** shows that linguistic information (**POS and NE**) helps improve performance significantly (CNN-based model improves most from 81.83% to 88.21%). The improvement mainly comes from the linguistic information attached to OOVs (the training set OOV ratio[3] is over 40%; the development and test sets are over 30%). We observe that without the attached POS and NE information, OOVs provide no information useful for identifying the numeral category.

Performance incorporating handcrafted category patterns: The third row in **Table 5** shows the performance when incorporating category-pattern features (extracted by heuristic rules). In comparison with the performance gained from linguistic knowledge, category-pattern features offer only small improvements, or even degrade performance. This may be because our simple manually-encoded rules do not cover enough patterns, as these patterns are quite diverse.

Performance with pre-processing[4] for numeral-splitting. Table 5 shows that automatically detecting local patterns via CNN filters delivers unsatisfactory performance. We suspect this is due to the task's high OOV rate. As most OOVs are concatenations of a numeral and other symbols, we split each token with numbers into individual sub-tokens. For example, "$80" is split to "$" and "80" two sub-tokens and '11/09/17' becomes '11', '/', '09', '/', and '17' five sub-tokens. After numeral splitting, OOV ratios[5] were reduced to 25%, 22%, and 23% on the training, development, and test sets, respectively. We then retrained all models with and without POS and NE information. **Table 6** shows that the micro F-measures of CNN, RNN, and

[3] The OOV ratio, obtained from the results of data preprocessing, is based on the 840B.300d pre-trained embeddings.

[4] Conducted after submission, the results of this experiment are not shown in the official evaluation results.

[5] The OOV ratio is obtained based on 840B.300d pre-trained embeddings.

Table 6. Test set performances with numeral-splitting preprocessing. "**None**" denotes the NN model without any knowledge. "**POS&NE**" denotes the NN models that incorporate both POS and NE information.

Knowledge	CNN		RNN		RNN with CNN filter	
	Micro	Macro	Micro	Macro	Micro	Macro
None	89.56%	83.17%	92.27%	86.60%	92.11%	88.18%
+ POS&NE	90.68%	83.60%	91.95%	88.36%	92.99%	88.25%

RNN+CNN models gained 7.7% (from 81.83% to 89.56%), 8.1% (84.22% to 92.27%), and 9.4% (82.71% to 92.11%), respectively, even without POS and NE information. With both POS and NE, the RNN+CNN model performance improved by 3.27% (89.72%[6] to 92.99%). These results show that the category patterns automatically learned by CNN outperform the handcrafted patterns of **Table 5**.

4 Conclusion and Future Work

We propose various ways to incorporate linguistic features into DNN models to classify numerical categories for the *FinNum* tasks of NTCIR-14. The results show that linguistic features improve performance. Furthermore, we show that suitable pre-processing (i.e., splitting numerals in the Twitter string in advance) to reduce OOV rates also significantly improves performance. Joint use of both approaches could offer additional benefits.

Although the numeral-splitting process helps CNN to learn better patterns, it may lose original concatenation information. For example, the '4' in 'P4K' and 'P 4 K' actually represent two different concepts (the former represents a product-number category, but the latter represents a quantity category). Furthermore, some keywords could help us to identify the numerical category. For example, '50' will be identified as an 'Indicator' by the keyword 'RSI' in the sentence 'Need RSI above 50.'. We will address these issues to further improve performance in the future.

References

1. Chen, C.C, Huang, H. H., Takamura, Hiroya and 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).
2. Chen, C. C., Huang, H. H., Shiue, Y. T., and Chen, H. H. 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).
3. Shapiro, A. M. How including prior knowledge as a subject variable may change outcomes of learning research. *American Educational Research Journal*, *41*(1), 159-189. (2004).
4. Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332-1338. (2015).

[6] The best performance in **Table 5**.

10

5. Huang, Z., Xu, W., and Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*. (2015).
6. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60). (2014)
7. Pennington, J., Socher, R., and 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).
8. Kim, Y. (2014). Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). (2014).
9. Chiu, J. P. and Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357-370. (2016).
10. Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao Y., Gao, Q., Macherey K., Klingner J., Shah A., Johnson M., Liu X. Kaiser Ł., Gouws S., Kato Y., Kudo T., Kazawa H., Stevens K., Kurian G., Patil N. Wang W., Young C., Smith J., Riesa J., Rudnick A., Vinyals O., Corrado G., Hughes M. and Dean J. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. (2016).