DeepMRT at the NTCIR-14 FinNum Task: A Hybrid Neural Model for Numeral Type Classification in Financial Tweets

Qianhui Wu*[1], Guoxin Wang*[2], Yuying Zhu[3], Haoyan Liu[4], and Börje F. Karlsson[2]

[1] Department of Automation, Tsinghua University
[2] Microsoft Research Asia
[3] Department of Computer Science, Nankai University
[4] State Key Lab of Software Development Environment, Beihang University
wu-qh16@mails.tsinghua.edu.cn, {guow,Borje.Karlsson}@microsoft.com, yuyzhu@mail.nankai.edu.cn, haoyan.liu@buaa.edu.cn

Abstract. Numerals contain much information in the financial domain and thus playing a crucial role in financial analysis processes. In this paper, we focus on the type classification task of numerals in financial tweets and propose a hybrid neural model.
A mention model employs a multi-layer perceptron to extract information from target numerals, while a context model utilizes recurrent neural networks to encode preceding and post context separately.
Moreover, we present several feature templates to replace inputs like pre-trained word vectors, which help the model handle problems caused by sparse numeral embeddings. Experimental results demonstrate that the proposed approach well outperforms baseline methods.

Keywords: Numeral type classification · Fine-grained entity type classification · Financial tweets · NTCIR-14-FinNum.
Team Name: DeepMRT
Subtasks: Subtask 1 and Subtask 2

1 Introduction

There has been a surge of interest in applying natural language processing (NLP) technologies to the financial domain over the past few years, leading to many successfully held workshops and shared tasks such as SemEval 2017 Task 5 [7], ECONLP 2018 [10], and FinNum2019 [5]. Numerals contain much information in financial texts and thus playing a crucial role in the financial analysis. For example, both stock price and price-earnings ratio (P/E ratio) of a given company have a direct effect on decision-making behavior of investors. Accordingly, we focus on the entity type classification task of numerals in financial tweets in this paper.

* These authors contributed equally to this work.

Entity type classification aims to assign type labels to entity mentions according to their context. Compared with Named Entity Recognition (NER), a traditional entity type classification task concerning standard coarse types such as PERSON and LOCATION in [12] and [22], this paper belongs to the realm of fine-grained entity type classification (FETC), which classifies entity mentions to a wider-coverage set of labels such as 17 types in [4] and 22 types in [9].

Early FETC systems mostly leverage conventional machine learning methods such as linear classifier perceptron [14] and SVMs [26]. Later, [1], [18], and [25] propose to map both features and labels to the same low-dimension space such that each entity is close to the corresponding label, achieving significant performance improvement. Recently, recurrent neural network (RNN) and its variants have brought great success to NLP tasks. Therefore, some works employ RNN, especially long-short-term-memory (LSTM), to learn representations for given entities and their context [9, 19, 20, 24, 27]. For example, [24] uses bidirectional LSTM to extract context information and demonstrates its effectiveness. However, especially when targeting numerals in financial tweets, the aforementioned methods can face limitations due to the extreme sparsity of numeral embeddings - a specific numeral may appear very few times in the dataset such that its word embedding cannot express the semantics exactly and sufficiently.

As for numeral information extraction, many existing works are rule-based, such as NumberRule [16]. Nonetheless, compared with news and other formal text forms, social media data like tweets here are much more challenging to analyze because of their informal and flexible writing style, potentially affecting the performance of rule-based systems. In addition, [4] proposes a CNN-based model and a RNN-based model, both of which take the matrix encoded from a target numeral and its context information as the input.

In this paper, to work around sparse numeral embeddings, we design several feature templates to generate informative input features. With the view that both numerals and their context play important roles in determining numeral types, but the complexity of learning to represent each is distinct, we use different networks to model numerals and context, specifically, two LSTMs for both pre- and post- context and one Multi-Layer Perceptron (MLP) to target numerals themselves. Extensive experimental results show that our approach well outperforms baseline methods.

2 Related Work

Named Entity Recognition. Named Entity Recognition (NER) is a traditional entity typing task that typically focuses on a small set of coarse types like PERSON, ORGANIZATION, LOCATION, etc. Traditional NER systems mostly focus on conventional machine learning methods, such as the hidden Markov model (HMM) [2] and conditional random field (CRF) [17]. More recently, NER systems have achieved significant improvements with the usage of deep neural network (DNN). [11] first uses BiLSTMs at the word-level to extract both pre- and post- context information from sentences. [12] and [6] incorporate character-level

NTCIR-14 Conference: Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies, June 10-13, 2019 Tokyo Japan

information in their proposed model structures to achieve additional improvements. Lately, [22] proposes stacked BiLSTM models using residual connections to alleviate the degradation problem of DNNs; and [15] employs an LM-LSTM-CRF architecture to co-train the NER model with a task-aware language model, aiming to promote character-level information extraction.

Fine-Grained Entity Type Classification. Compared with NER, fine-grained entity type classification (FETC) expands coarse-grained types into a wider set of fine-grained types, typically including sub-types. Early works regard FETC as a multi-label classification task and utilize linear classifier perceptron [14] as well as multiple binary SVMs [26]. Later, some approaches propose to map both features and labels to a low-dimension space, where objects with similar types are supposed to be closer, whereas mutually exclusive labels are further apart [1, 18, 25], while others propose to learn representations for given entities and their context, and use an output layer such as softmax for the entity type classification decision [9, 19, 20, 24, 27]. For the former, [18] employs a joint mention-type model for mapping, while [25] constructs the model based on WSABIE [23]. As for the later, [9] leverages RNN and MLPs to learn mention and context representations respectively. [19] and [20] utilize BiLSTM-based context models and introduce an attention mechanism. [24] further incorporates LSTM and an average operator to promote mention representation.

Numeral Information Extraction. Numerals and their semantics play important roles in various scenarios such as question answering [8], medical documents analysis [21], and financial text analysis [4]. [8] makes use of relation-defining patterns and WordNet similarity information to extract and approximate numerical information such as height and weight from the Web. [13] proposes a temporal information extractor with probabilistic inference. [16] presents a rule-based system and a probabilistic graphical model to extract numeral relation with minimal human supervision requirement. And [21] deals with temporal relation extraction over clinical notes based on in depth feature engineering. Aiming at fine-grained numeral type classification in financial tweets, [4] encodes a target numeral and its context as a matrix, and further employs CNN and RNN to extract higher-level information.

This paper focuses on understanding numerals in financial tweets as in [4]. Compared with [4], we achieve significant performance improvement based on feature design and representation learning of both mentions and their context. Compared with other aforementioned works dealing with entity recognition or type classification, our approach is distinct due to the extreme sparsity of numeral embeddings.

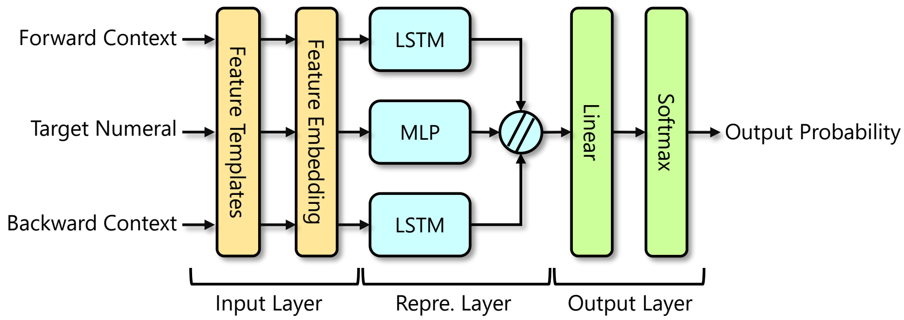

Fig. 1. Main architecture of the proposed model - DeepMRT.

3 Model

To begin with, we formulate the numeral type classification problem as follows. Given an input sentence $\mathbf{s}$, we first tokenize the sentence into N tokens, i.e., $\mathbf{s} = \{w_1, \ldots, w_i, \ldots, w_N\}$ where w_i is the i-th token. The aim of this paper is to predict the type of each numeral in $\mathbf{s}$. Assuming that w_t is a numeral (digits with a decimal separator), we compute the corresponding distribution of w_t, i.e., $y \in \mathbb{R}^{C \times 1}$ for the C types, and regard the type with the largest probability as the predicted label.

As illustrated in Fig. 1, the proposed model consists of three layers, i.e., an input layer, a representation layer, and an output layer. We will elaborate on each of them in the following subsections.

3.1 Input layer

Pre-trained word embeddings are widely used in existing NLP tasks. However, the mention forms focused on in this paper are numerals, whose embeddings are too sparse and inaccurate to express their semantics. Therefore, we present several feature templates to generate high-quality input features. The input layer focuses on mapping these generated discrete features into distributed feature embeddings.

Feature Templates. General linguistic, orthographic, and morphological features (like prefixes or suffixes of a word) are utilized in our system. Additionally, features generated from the Brown Cluster [3], which assigns words to classes based on the frequency of their co-occurrence with other words, and from specialized extractors in Microsoft Recognizers-Text[5] are also leveraged to capture deeper semantic information. The detail of the feature templates are listed as follows:

[5] https://github.com/Microsoft/Recognizers-Text

- **Normalization -** w_i^n**:** We lowercase w_i and replace each digit, if existing, with 0.
- **Orthography -** w_i^p**:** We replace each character in w_i with an encoding corresponding to certain patterns. The patterns and corresponding encoding used here consist of `UPPERCASE` (C), `LOWERCASE` (c), `NUMERAL` (n), `PUNCTUATION` (p), `PERIOD` (.), and `OTHER` (o).
- **Format -** w_i^f**:** We use a string to denote the format of w_i. These are based on pattern matching and include: `INTEGER`, `REAL`, `FRACTION`, `TIMESCORE`, `NUM-WORD`, `NUM-NUM`, `COMPOUND`, `NUM-NO`, `YEAR`, `UPPERCASE_S`, `LOWERCASE_S`, `NUMBER_S`, `OTHER_S`, `UPPERCASE`, `LOWERCASE`, `CAPITALIZATION`, `MIX-UPPER-LOWER`, `MIX-UPPER-LOWER-NUMBER`, `MIX-ALL`, and `OTHER`; where `_S` refers to single-letter tokens.
- **Pre-/suffix -** $w_i^{:j}$**:** The j-character pre-/suffix of w_i.
- **Brown Cluster -** $c_i^{:j}$**:** The j-character prefix of the Brown cluster c_i.
- **Recognizers-Text Type -** $w_i^{(j)}$**:** The label of w_i on the j-th class, annotated by Microsoft.Recognizers.Text. We use four classes of labels for each token as external features here: `NUMBER`, `PERCENTAGE`, `CURRENCY`, and `DATE/TIME` - where `DATE/TIME` can be further divided into six sub-types, i.e., `DATETIME`, `DATERANGE`, `DATE`, `TIME`, `DURATION`, and `SET`.

Feature Embedding. For each feature, the embeddings are mapped from a randomly initialized look-up table $E(\cdot)$. Taking the normalization feature w_i^n as an example, the corresponding feature embedding is formulated as:

$$x_i^n = E^n(w_i^n) \tag{1}$$

To sum up, given an input token w_i, the output of the input layer, denoted as x_i, is the concatenation of all feature embeddings.

3.2 Representation Layer

The representation layer aims to extract high-level semantic features for both target numerals and their context tokens. Here, we learn numeral representations $h^{(n)}$ and context representations $h^{(c)}$ with different models. For a target numeral w_t, the layer output, which is denoted as h_t, is the concatenation of both representations:

$$h_t = [h_t^{(n)}; h_t^{(c)}] \tag{2}$$

Numeral Representation. Given the feature embedding x_t of a target numeral w_t, we apply a multi-layer perceptron (MLP) with one hidden layer utilizing ReLU activation to calculate the numeral representation $h_t^{(n)}$:

$$h_t^{(n)} = \mathrm{ReLU}(W_t x_t + b_t) \tag{3}$$

where W_t and b_t are trainable parameters.

Context Representation. The context representation is divided into two parts, the preceding context and the posterior context. Considering that the type of a numeral might only depend on either context, both context information should be distinguished.

Given feature embeddings of the preceding terms $\{x_1, x_2, \ldots, x_{t-1}\}$ and following terms $\{x_N, x_{N-1}, \ldots, x_{t+1}\}$, we apply two LSTMs to encode them separately:

$$h_t^{\rightarrow} = \mathrm{LSTM}^{\rightarrow}(x_1, x_2, \ldots, x_{t-1}) \tag{4}$$

$$h_t^{\leftarrow} = \mathrm{LSTM}^{\leftarrow}(x_N, x_{N-1}, \ldots, x_{t+1}) \tag{5}$$

where $h_t^{\rightarrow}$ denotes the preceding context representation, while $h_t^{\leftarrow}$ denotes the posterior context representation. The final context representation $h_t^{(c)}$ is their concatenation, which can be formulated as:

$$h_t^{(c)} = [h_t^{\rightarrow}; h_t^{\leftarrow}] \tag{6}$$

3.3 Output Layer

We employ a linear function followed by a softmax function over the representation layer to compute probabilities for C types:

$$p(y|h_t) = \mathrm{softmax}(W_o h_t + b_o) \tag{7}$$

where W_o and b_o are trainable parameters.

Training. The loss function used here is defined as the following negative log-likelihood:

$$\mathcal{L} = -\log p(y^*|h_t) \tag{8}$$

where y^* is the given label of the target numeral w_t.

Inference. As for type classification, we choose the label with the maximal probability as the predicted type $\hat{y}$:

$$\hat{y} = \arg\max_{y}\ p(y|h_t) \tag{9}$$

4 Experiments

4.1 Task Definition

In the experiments, we conduct two sub-tasks separately. The definitions of the sub-tasks [5] are shown as follows:

Subtask 1: Classify the numerals in the tweets into 7 categories, which are `Monetary`, `Percentage`, `Option`, `Indicator`, `Temporal`, `Quantity`, and `Product/Version Number`.

Subtask 2: Classify the numerals at the subcategory level, which has 17 classes and is associated with categories in Subtask 1. All categories are shown in Table 1.

<table>
<tr><th>Category</th><th>Subcategory</th><th>Category</th><th>Subcategory</th></tr>
<tr><td rowspan="9">Monetary</td><td>money</td><td rowspan="2">Percentage</td><td>relative</td></tr>
<tr><td>quote</td><td>absolute</td></tr>
<tr><td>change</td><td rowspan="2">Option</td><td>exercise price</td></tr>
<tr><td>buy price</td><td>maturity date</td></tr>
<tr><td>sell price</td><td>Indicator</td><td></td></tr>
<tr><td>forecast</td><td rowspan="2">Temporal</td><td>date</td></tr>
<tr><td>stop loss</td><td>time</td></tr>
<tr><td rowspan="2">support or resistance</td><td>Quantity</td><td></td></tr>
<tr><td>Product</td><td></td></tr>
</table>

Table 1. (Sub)Categories

4.2 Data

We use the dataset provided by the FinNum 2019 task (co-located with NTCIR-14) [4], which has already been divided into training, development, and test splits. This dataset was constructed from financial tweets and was annotated by three experts with financial domain knowledge. A total of 7 categories are proposed for numerals, and four of them are further extended to various subcategories. Table 2 shows the statistics of the dataset.

Dataset	Sentences	Numerals
train	4.1k	6.9k
test	0.79k	1.3k
dev	0.46k	0.75k

Table 2. Statistics of the dataset.

4.3 Preprocessing Procedure

We first build a rule-based tokenizer to locate all numerals in a sentence. Because a financial tweet may contain more than one numeral in the text, we separate them into independent queries, in other words, classifying one numeral at a time. We also divide the sentence into three parts, i.e., the target numeral, its preceding tokens, and its posterior tokens. For example, the tweet "I am bearish on $RACE with a target price of $46 in 6 mos. on Vetr!" which includes two numerals, i.e., 46 and 6, will be separated into two queries as shown in Table 3. To further reduce the noise from the financial tweets, we replace all URLs with a `<URL>` token.

4.4 Experimental Settings

We use micro- and macro-averaged F-scores to evaluate the overall performances of our model over both subtask 1 and subtask 2. In the experiments, we do not

Given tweet	I am bearish on $RACE with a target price of $46 in 6 mos. on Vetr!	
Query 1	target numeral	46
	preceding tokens	I am bearish on $ RACE with a target price of $
	posterior tokens	in 6 mos . on Vetr !
Query 2	target numeral	6
	preceding tokens	I am bearish on $ RACE with a target price of $ 46 in
	posterior tokens	mos . on Vetr !

Table 3. An example of the preprocessing procedure.

use pre-trained word embeddings. For the hyper-parameters of feature templates, we use $j \in \{2, 3\}$ for prefixes, $j \in \{2, 3, 4\}$ for suffixes, and $j \in \{4, 6, 8, 10, l\}$ for Brown Clusters, where l represents the string length. We set the embedding size for each feature according to the total number of all possible values of it. In the experiment not using Recognizer-Text Type features, the summed dimension of all features is 163, while in the experiment using them, the total dimension is 173. The hidden size is set to 100 for the LSTM and 80 for the DNN. Dropout is applied to both LSTM and DNN layer with a rate of 0.4; epoch and batch size are set to 24 and 2, respectively. Adam is used for optimization, with an initial learning rate of 0.001.

4.5 Experimental Results and Discussions

Table 4 shows the F-scores (%) of the experimental results, including both micro- and macro-averaged F-scores. The top section of the table show the results from [4] as baselines. The bottom section of the table shows the results from our proposed model with different type of features. Basic Features in the table represents the set of features including Normalization, Orthography, Format, Prefix/Suffix, and Brown Cluster.

Model	**Sub-task1**		**Sub-task2**	
	Micro	Macro	Micro	Macro
[Chen *et al.*, 2018][6] [4]	54.90	51.67	43.75	31.12
Basic Features	91.16	84.72	81.27	75.59
+ Recognizers-Text Type	**91.87**	**87.94**	**83.03**	**77.90**

Table 4. Results on the FinNum 2019 dataset.

In Subtask 1, our model achieves macro-averaged F-scores of 87.94% and micro F-scores of 91.87%, which significantly outperforms the baseline, which is 54.90% on micro F-scores and 51.67% on macro-averaged F-scores, produced by a word-based CNN model.

[6] The experimental results in the source paper are obtained from Finnum 1.0, and the results here are from the authors' reproduction on Finnum 2.0.

In Subtask 2, for the fine types numeral entity classification, the baseline result is produced by char-based CNN, which is micro F-scores of 43.75% and macro-averaged F-scores of 31.12%. Our model achieves micro F-scores of 81.27% and macro-averaged F-scores of 75.59% with basic features, almost 50% improvement compared with the previous state-of-the-art method, demonstrating the effectiveness of our proposed model structure. Adding the Recognizers-Text Type features results in significant additional improvement.

The experimental results of both subtasks demonstrate the effectiveness of the Recognizer-Text types features on top of our proposed model.

5 Conclusion

We introduce a hybrid neural model to deal with type classification of numerals in financial tweets. Considering that both numerals and their contexts play important, but different, roles in determining numeral types, we apply an MLP to learn numeral representations an two LSTMs to learn context representation, respectively. Moreover, to cope with the extreme sparsity of numeral embeddings, we design several feature templates to generate informative input features instead of directly taking pre-trained word embeddings as inputs. Experimental results illustrate that the proposed approach achieves better performance compared with the baseline methods.

References

1. Abhishek, A., Anand, A., Awekar, A.: Fine-grained entity type classification by jointly learning representations and label embeddings. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers (2017)
2. Bikel, D.M., Miller, S., Schwartz, R., Weischedel, R.: Nymble: A high-performance learning name-finder. In: Proceedings of the Fifth Conference on Applied Natural Language Processing. pp. 194–201 (1997)
3. Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Computational linguistics **18**(4), 467–479 (1992)
4. Chen, C.C., Huang, H.H., Shiue, Y.T., 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)
5. 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)
6. Chiu, J., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics **4**, 357–370 (2016)
7. Cortis, K., Freitas, A., Daudert, T., Huerlimann, M., Zarrouk, M., Handschuh, S., Davis, B.: Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 519–535 (2017)

8. Davidov, D., Rappoport, A.: Extraction and approximation of numerical attributes from the web. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 1308–1317 (2010)
9. Dong, L., Wei, F., Sun, H., Zhou, M., Xu, K.: A hybrid neural model for type classification of entity mentions. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
10. Hahn, U., Hoste, V., Tsai, M.F.: Proceedings of the first workshop on economics and natural language processing. In: Proceedings of the First Workshop on Economics and Natural Language Processing (2018)
11. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR (2015), http://arxiv.org/abs/1508.01991
12. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 260–270 (2016)
13. Ling, X., Weld, D.S.: Temporal information extraction. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
14. Ling, X., Weld, D.S.: Fine-grained entity recognition. In: Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)
15. Liu, L., Shang, J., Xu, F.F., Ren, X., Gui, H., Peng, J., Han, J.: Empower sequence labeling with task-aware neural language model. CoRR **abs/1709.04109** (2017), http://arxiv.org/abs/1709.04109
16. Madaan, A., Mittal, A., Ramakrishnan, G., Sarawagi, S., et al.: Numerical relation extraction with minimal supervision. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
17. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 (2003)
18. Ren, X., He, W., Qu, M., Huang, L., Ji, H., Han, J.: Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016)
19. Shimaoka, S., Stenetorp, P., Inui, K., Riedel, S.: An attentive neural architecture for fine-grained entity type classification. In: Proceedings of the 5th Workshop on Automated Knowledge Base Construction (2016)
20. Shimaoka, S., Stenetorp, P., Inui, K., Riedel, S.: Neural architectures for fine-grained entity type classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers (2017)
21. Tourille, J., Ferret, O., Tannier, X., Névéol, A.: Temporal information extraction from clinical text. In: Conference of the European Chapter of the Association for Computational Linguistics (2017)
22. Tran, Q., MacKinlay, A., Jimeno Yepes, A.: Named entity recognition with stack residual lstm and trainable bias decoding. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 566–575 (2017)
23. Weston, J., Bengio, S., Usunier, N.: Wsabie: Scaling up to large vocabulary image annotation. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)
24. Xu, P., Barbosa, D.: Neural fine-grained entity type classification with hierarchy-aware loss. arXiv:1803.03378 (2018)

25. Yogatama, D., Gillick, D., Lazic, N.: Embedding methods for fine grained entity type classification. 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) (2015)
26. Yosef, M.A., Bauer, S., Hoffart, J., Spaniol, M., Weikum, G.: Hyena: Hierarchical type classification for entity names. In: Proceedings of COLING 2012: Posters. pp. 1361–1370 (2012)
27. Zhang, S., Duh, K., Van Durme, B.: Fine-grained entity typing through increased discourse context and adaptive classification thresholds. In: Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (2018)