KCAT: A Knowledge-Constraint Typing Annotation Tool

Sheng Lin[1], **Luye Zheng**[1], **Bo Chen**[1], **Siliang Tang**[1]*, **Yueting Zhuang**[1], **Fei Wu**[1], **Zhigang Chen**[2], **Guoping Hu**[2] **& Xiang Ren**[3]
[1]Zhejiang University
[2]iFLYTEK Research, [3]University of Southern California,
{shenglin, antlar, chenbo123}@zju.edu.cn,
{siliang, yzhuang, wufei}@zju.edu.cn,
{zgchen, gphu}@iflytek.com,
xiangren@usc.edu

Abstract

Fine-grained Entity Typing is a tough task which suffers from noise samples extracted from distant supervision. Thousands of manually annotated samples can achieve greater performance than millions of samples generated by the previous distant supervision method. Whereas, it's hard for human beings to differentiate and memorize thousands of types, thus making large-scale human labeling hardly possible. In this paper, we introduce a Knowledge-Constraint Typing Annotation Tool (KCAT[1]), which is efficient for fine-grained entity typing annotation. KCAT reduces the size of candidate types to an acceptable range for human beings through entity linking and provides a Multi-step Typing scheme to revise the entity linking result. Moreover, KCAT provides an efficient Annotator Client to accelerate the annotation process and a comprehensive Manager Module to analyse crowdsourcing annotations. Experiment shows that KCAT can significantly improve annotation efficiency, the time consumption increases slowly as the size of type set expands.

1 Introduction

Recent years Natural Language Processing community has seen a surge of interests in fine-grained entity typing (**FET**) as it serves as an important cornerstone of several nature language processing tasks including relation extraction (Mintz et al., 2009), entity linking (Raiman and Raiman, 2018), and knowledge base completion (Dong et al., 2014). Given an entity mention (i.e. a sequence of token spans representing an entity) in the corpus, **FET** aims at uncovering its context-dependent type. Table 1 includes Fine-grained Entity Typing datasets in recent years, the target types often form a type hierarchy.

The difficulty of FET and FET Annotation both increase rapidly with the growth of type hierarchy's depth. Previous research work mainly focus on generating train corpus with distant supervision (Ling and Weld, 2012; Gillick et al., 2014; Ren et al., 2016a; Choi et al., 2018). In spite of its efficiency, distant supervision brings the problem of noisy labels, for example, {*Other, brand*} are noisy labels for *'Kobe'* in *"Kobe scored 60 points in the final game."*. According to (Choi et al., 2018), 6000 manually labeled samples achieved greater performance than millions of samples generated by distant supervision. (Onoe and Durrett, 2019) observed that noisy samples may even cause damage to the performance of FET model. (Ren et al., 2016b; Onoe and Durrett, 2019) proposed label noise reduction methods, which are pretty complicated and hard to migrate.

Thus the annotation corpus for FET is important and necessary. However, it is not easy to annotate a corpus for FET since it's hard for human beings to differentiate and memorize thousands of types.

To solve this extremely hard annotation task, we use Entity Linking (EL) to constrain the candidate types of the entity mention. Entity Linking, which tries to link entity mention to a unique entity in a specific knowledge base (i.e. Yago or Freebase), has been studied for years. The state-of-the-art EL system yields 0.93 F1 score in Conll2003(Sang and Buchholz, 2000), while the F1 scores of FET vary from 0.40 (Onoe and Durrett, 2019) to 0.79 (Abhishek et al., 2017) on different datasets. With the help of EL, the candidate types of a mention can be greatly reduced as shown in the Table 1. Based on this observation, we develop a Knowledge-Constraint Typing Annotation Tool (KCAT). KCAT uses an external entity linking tool to constrain the candidate types of

*Corresponding Author.
[1]Code is available at https://github.com/donnyslin/KCAT

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 99–104
Florence, Italy, July 28 - August 2, 2019.

Dataset	Depth	#Types	#KC Types	Ratio.%
BBN	2	47	2.1	4.3%
FIGER	2	110	2.3	2.1%
TAC 2018	14	7309	8.5	0.1%

Table 1: Size of Candidate Types before and after **Knowledge-Constraint** on Different Datasets, and the ratio of the latter to the former

mentions. Because errors made by Entity Linking are inevitable, we provide an EL revision extension in KCAT which can also help the annotation of Entity Linking. The extension uses the coarse-grained type of mention to constrain the candidate entities of mention, which greatly saves the time of revising EL result. The details of the annotation interaction are described in section 3. Besides using the Knowledge Constraint technology to make the annotation of FET easier, KCAT provides other 4 functions to further improve the annotation efficiency: Hierarchical Structure Visualization, Annotation Hint, Annotation Modification and Annotation Export. In brief, KCAT has the advantages as follows:

- **Knowledge-Constraint:** it visualizes candidate types hierarchically, which is extracted from Knowledge Base and reduced by entity linking.
- **Efficient:** it supports multiple shortcuts to improve annotation efficiency; entity description of the wiki and type description to help distinguish candidate types.
- **Portable:** it is only required to replace a few json files to complete the migration of different datasets.
- **Comprehensive:** it supports crowdsourcing results comparison and integration.

The rest of the paper is organized as: Section 2 briefly describes recent research in FET. Section 3 introduces the overview of our framework. Section 4 describes the architecture of KCAT and its detail functions. Section 5 analyses the efficiency comparison and annotation quality in different annotation mode. Finally, Section 6 concludes this paper.

2 Related Works

Named Entity Recognition (NER) has been studied for several decades, which classifies coarse-grained types (e.g. person, location). In order to reduce the cost of obtaining fine-grained typing corpus, distant supervision has been widely used in FET (Ling and Weld, 2012; Gillick et al., 2014; Ren et al., 2016a; Choi et al., 2018). Inevitably, distant supervision brings the unique challenge of noisy labels in FET which seriously slows down the research process in this field. Many researchers focus on noise reduction of label(Ren et al., 2016b,a; Onoe and Durrett, 2019). The costliness of annotated corpus and the problem of noisy labels greatly hurt the usability of FET technique. Previous entity typing annotation tools (Stenetorp et al., 2012; Yang et al., 2018) focus on the coarse-grained types and is hard to migrate to fine-grained types. A specific and carefully designed annotation tool is urgently needed for FET. To the best of our knowledge, KCAT is the first Fine-grained Entity Typing Annotation tool.

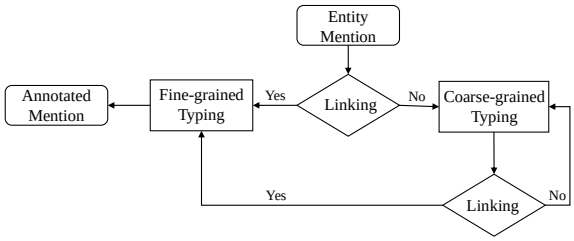

Figure 1: The framework of KCAT

3 Overview

This section overviews the proposed framework as shown in Figure 1. KCAT leverages **Type Hierarchy from Knowledge Base**, and reduces the size of candidate types to a small range through **Entity Linking**. Furthermore, KCAT proposes a **Multi-step Typing** scheme because the result from Entity Linking may be incorrect. As shown in Figure 1, given an entity mention, KCTA links it to entity in Knowledge Base by EL model. Fine-grained type can be directly labeled if this result from model is correct, otherwise KCTA provides entity linking revision by coarse-grained type constraint to filter out candidates entities with inconsistent types and finally labels fine-grained type. The details will be described following.

3.1 Type Hierarchy and Knowledge Base

Given a set of types $\mathcal{T} = \{t_1, ..., t_N\}$, these types usually form a Directed Acyclic Graph (DAG) or more commonly a tree. Each entity e in Knowledge Base $\mathcal{K}$ has only several types, $\mathcal{T}_e = \{t_1, ..., t_M\} \subset \mathcal{T}$. In general, the size $|\mathcal{T}_e|$ is far

less than the size $|\mathcal{T}|$. As shown in Table 1, the average size $\overline{|\mathcal{T}_e|}$ maintains in a small range as $|\mathcal{T}|$ expands. Therefore, EL, as a Knowledge Constraint method, helps to reduce the candidate size significantly.

3.2 Entity Linking

Given a set of entity mentions $\mathcal{M} = \{m_1, ..., m_T\}$ in corpus $\mathcal{D}$, Entity Linking aims to link each mention m_t to its corresponding gold entity e_i^* in $\mathcal{K}$. Such process is usually divided into two steps: *Candidate generation* first collects a set of possible candidate entities $\mathcal{E}_i = \{e_i^1, ..., e_i^{|\mathcal{E}_i|}\}$ for m_i; *Candidate ranking* is then applied to rank all candidates. The linking system selects the top ranked candidate as the predicted entity $\hat{e}_i$. Given a mention in text, our system firstly links it to $\mathcal{K}$ by state-of-the-art Entity Linking system (Le and Titov, 2018), which yields 0.93 F1 score on Conll 2003 dataset. As shown in Figure 2, *Kobe* is an entity mention which can be linked to "Kobe Bean Bryant" in $\mathcal{K}$, whose types only contain *person* and its descendant, we only need to type on the subtree. Whereas, after EL there are still 7% wrong linked mentions which need to be manually revised. Hence, for each mention, we provide the candidate entities with top 20 ranked score from EL system as the revision choices. In current EL systems, ground truth entity coverage can reach 0.98 (Ganea and Hofmann, 2017) which ensuring the recall of revision choices.

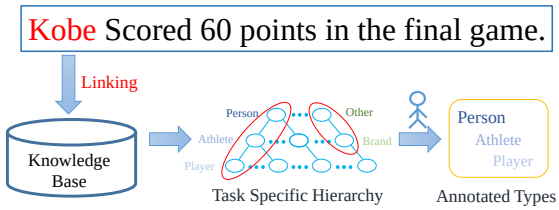

Figure 2: The Process of Knowledge Constraint by Entity Linking

Even though, the revision progress can be tough as The context may be ambiguous for manually linking mentions. Distinguishing different entities can be time-consuming and difficult for an amateurish annotator. To accelerate the revision progress, KCAT uses multi-step typing to reduce the number of candidate entities.

3.3 Multi-step Typing

Entity Linking and **Entity Typing** are mutually improved: a) EL helps to reduce the size of candidate types of ET; b) ET helps to filter out irrelevant candidate entities of EL with inconsistent types. Based on this observation, we propose a Multi-step Typing scheme. Figure 3 demonstrates the interaction between entity typing and linking. The left part shows the candidate entities in every step and the right part shows the candidate types constrained by candidates entities. The red words"*Liverpool*" is an entity mention "*Liverpool*", EL mistakenly links football *Club* "*Liverpool F.C.*" to "*Liverpool City*". It's tough for human to label the mention as *Club* without professional knowledge, but it's easy to label it as *Organization*. In our scheme, the user firstly selects the coarse-grained type *Organization*, and observes that the candidate entities which contain *Organization*. Gold entity, "Liverpool F.C.", and *Club* can be easily picked out. Through hierarchy type selection, the user focus on a few candidate entities, which can prompt user to do deeper type selection.

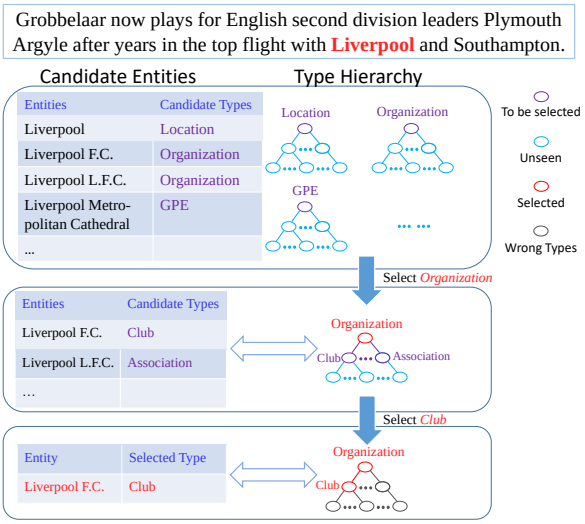

Figure 3: Interaction between Typing and Linking

4 KCAT

KCAT is developed based on standard Python GUI library Tkinter, hence it only needs Python installation as prerequisite. It provides user-friendly interfaces for annotators. KCAT contains two main modules: **Annotator Client** and **Manager Module**.

- **Annotator Client:** With the help of Knowledge Constraint, KCAT makes the impossible annotation of FET possible. An Annotation Client is designed to further accelerate the annotation

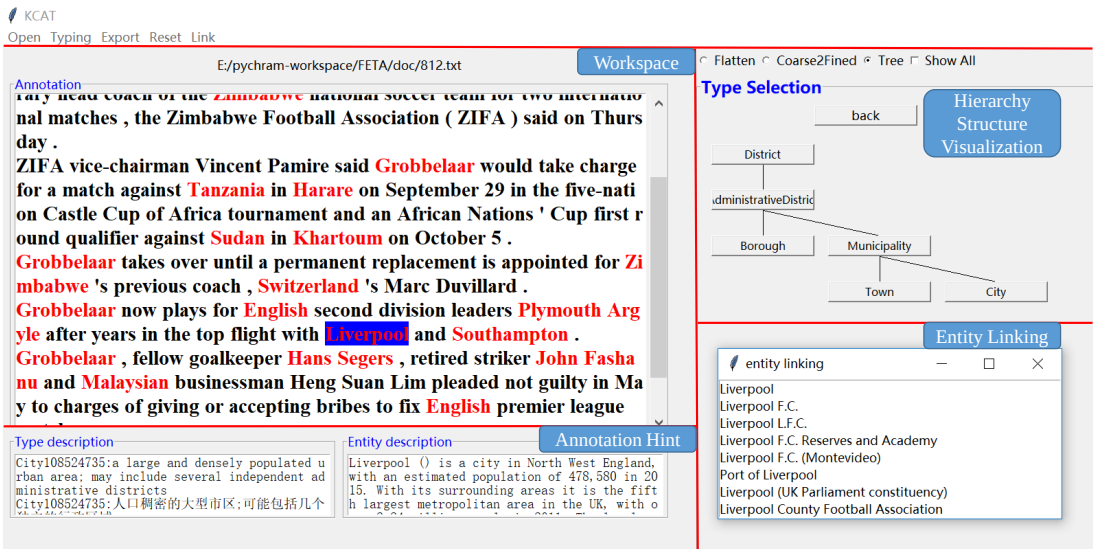

Figure 4: A Screenshot of KCAT (different parts are seperated by bold red line)

process. It provides 4 practical functions to reduce annotation time: **a) Hierarchical Structure Visualization**, **b) Annotation Hint**, **c) Annotation Modification**, **d) Annotation Export**.

- **Manager Module:** KCAT also provides a Manager Module to analyse the annotation results.

We will introduce these two modules in following sections respectively.

4.1 Annotator Client

Figure 4 shows the interface of Annotator Client. The interface consists of 5 parts. The toolbar contains some basic functions for **Annotation Modification** and **Annotation Export**. The main area in the upper left is the text in annotating, in which mentions are colored by red and types are colored by blue. The upper right area shows the **Hierarchical Structure Visualization** and the bottom left area shows the **Annotation Hint** which helps annotators to know more about the type and entity information. The children window titled as "entity linking" in the bottom right shows the candidate entities ranked by entity linking system.

a) Hierarchical Structure Visualization

On most entity typing datasets, types can be formed by hierarchical structure, hence Directed Acyclic Graph (DAG) is a good view to represent this structure. By this hierarchical structure, KCAT supports up-down searching without scanning irrelevant types in other subgraph.

b) Annotation Hint

Annotator may not memorize all the definitions of types and descriptions of entities. Therefore, prompting type definitions and entity descriptions will contribute to the labeling process. KCAT provides the definition of type extracted from WordNet(Fellbaum, 2012) and the first paragraph of Wikipedia page related to entity to help the annotator further acquainted with the type and entity.

c) Annotation Modification

Sometimes users need to revise the annotation when they make mistakes. KCAT provides several efficient modification actions to revise these incorrect annotations.

Action Undo and Redo: annotators can cancel their previous actions or redo their canceled actions to return to any previous states by press the shortcut key Ctrl+z and Ctrl+y respectively.

Label Modify and Reset: if an entity mention receives an incorrect type, annotator only needs to put the cursor inside the span and restart labeling. In addition, annotator can reset the annotations by shortcut key Ctrl+r.

d) Annotation Export

KCAT provides the "Export" function, which exports the annotated text as standard format (ended with txt) or json format so that it's easy to be processed by users.

4.2 Manager Module

The Manager Module aims to evaluate the quality of annotated files, analyzes the detailed disagreements of different annotators and integrates all annotation results from multiple annotators.

Multiple Annotators Comparison

The annotations from multiple annotators are inconsistent, in order to evaluate the quality of annotations, KCAT provides a Multiple Annotator Comparison interface to generate the accuracy matrix to measure consistency among multiple annotators, which is illustrated in Figure 5.

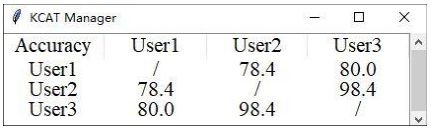

Figure 5: Multiple Annotator Analysis

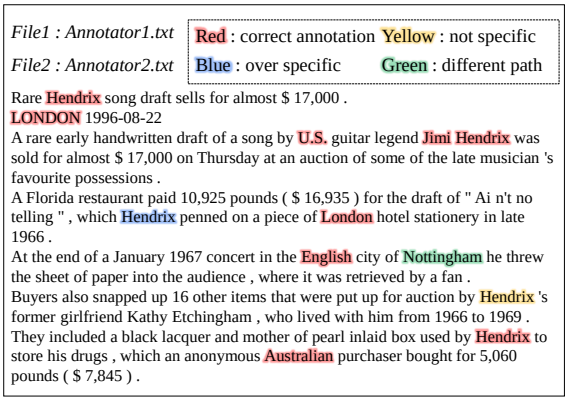

Figure 6: Error Analysis

Error Analysis

There are three common error patterns in entity typing task: (a) Over Specific; mislabeling parent type as child type (i.e., annotated as athlete while the ground truth type is person). (b)Not Specific; in contrary to former (a). (c)Incorrect Path; labeling the wrong child type (i.e., entity is annotated as athlete while the ground truth type is artist and they are both child type of person). KCAT provides an interface to generate the error analysis report in ".tex" format, as shown in Figure 6, different errors are rendered in different colors.

Annotations Integration

The Annotations Integration interface provides a method to integrate all annotation results from crowdsourcing annotations to generate final labels by voting.

5 Experiment

In order to verify the efficiency of KCAT, we conduct a mock annotation experiment. 100 sentences are extracted from the English dataset of Conll 2003 (Sang and Buchholz, 2000) as the corpus to be annotated. The entity mention spans in these sentences have been annotated. Type hierarchy is extracted from following three datasets: (1) Conll 2003; (2) BBN(Ren et al., 2016a); (3) FIGER(Ling and Weld, 2012); and YAGO Knowledge Base(Heng et al., 2018). The mappings between entity and its related types are provided by these datasets. We have chosen two annotation modes: (a) without pre-linking, directly through top-down search or flatten search; (b) filtering out types that are inconsistent with entity types through entity linking.

Annotation Efficiency. In Table 2, we compare the labeling time in aforementioned two modes and calculate the percentage of time saved with Entity Linking on different type set. It can be observed that with the number of types increases, time consumption increases slowly with entity linking, while without entity linking, time consumption increases exponentially. The percentage of time saved also increases as the number of types expands on different datasets. When there are thousands of types in Knowledge Base, such as YAGO Knowledge Base(Heng et al., 2018), which contains 7309 types, it's impossible for human to annotate.

Type Set	#types	depth	Time		
			w/ EL	w/o EL	Rel%
Conll	5	1	**5**	6	6.3%
BBN	47	2	**8**	17	52.9%
FIGER	112	2	**10**	30	66.7%
YAGO	7309	14	**13**	-	-

Table 2: Time Consumption (minute) of Annotating 60 Sentences on Different Datasets

Annotation Quality. Pairwise accuracy is used to measure the consistence between arbitrary two annotators. For multiple annotators, we can generate a heat map, each element of the heat map represents pairwise accuracy, darker color means higher accuracy. In Figure 7, the User 1, 2, and 3 adopt the mode (b), and the User 4, 5 and 6 adopts the mode (a), and it can be observed that the labeling quality of 1, 2, and 3 is significantly better than 4, 5, 6 as the former have higher consistency.

6 Conclusion

In this paper, we propose an efficient Knowledge Constraint Fine-grained Entity Typing Annotation Tool, which further improves entity typing process through entity linking together with some practical functions.

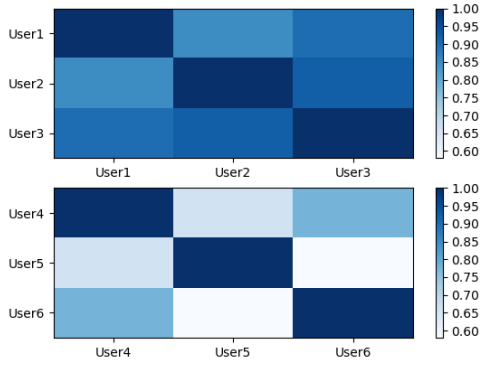

Figure 7: A Comparison of Multiple Annotators in Accuracy

7 Acknowledgement

This work has been supported in part by NSFC (No.61751209, U1611461), Zhejiang University-iFLYTEK Joint Research Center, Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Engineering Research Center of Digital Library, Ministry of Education. Xiang Ren's research has been supported in part by National Science Foundation SMA 18-29268.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. 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*, pages 797–807, Valencia, Spain. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. *arXiv preprint arXiv:1807.04905*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Christiane Fellbaum. 2012. Wordnet. *The Encyclopedia of Applied Linguistics*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Ji Heng, Sil Avirup, Trang Dang Hoa, J. Goldschen Alan, Duncan Jason, Getman Jeremy, Nothman Joel, Onyshkevych Boyan, Soboroff Ian, and Strassel Stephanie. 2018. Tac kbp2018 entity discovery and linking for 7,309 entity types. http://nlp.cs.rpi.edu/kbp/2018/EDL2018TaskSpec_V2.0.pdf.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Yasumasa Onoe and Greg Durrett. 2019. Learning to Denoise Distantly-Labeled Data for Entity Typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jonathan Raphael Raiman and Olivier Michel Raiman. 2018. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1825–1834. ACM.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. Yedda: A lightweight collaborative text span annotation tool.