METIC: Multi-Instance Entity Typing from Corpus*

Bo Xu[123], Zheng Luo[3], Luyang Huang[3], Bin Liang[35], Yanghua Xiao[36], Deqing Yang[4], Wei Wang[3]
[1]School of Computer Science & Technology, Donghua University, Shanghai, China
[2]CETC Big Data Research Institute Co.,Ltd., Guizhou, China
[3]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[4]School of Data Science, Fudan University, Shanghai, China
[5]Shuyan Technology, Shanghai, China
[6]Alibaba Group, Zhejiang, China
{xubo,zluo14,huangly14,liangbin,shawyh,yangdeqing,weiwang1}@fudan.edu.cn

ABSTRACT

This paper addresses the problem of *multi-instance entity typing from corpus*. Current approaches mainly rely on the structured features (*attributes*, *attribute-value pairs* and *tags*) of the entities. However, their effectiveness is largely dependent on the completeness of structured features, which unfortunately is not guaranteed in KBs. In this paper, we therefore propose to use the text corpus of an entity to infer its types, and propose a multi-instance method to tackle this problem. We take each mention of an entity in KBs as an instance of the entity, and learn the types of these entities from multiple instances. Specifically, we first use an end-to-end neural network model to type each instance of an entity, and then use an integer linear programming (ILP) method to aggregate the predicted type results from multiple instances. Experimental results show the effectiveness of our method.

CCS CONCEPTS

• **Computing methodologies → Information extraction**; **Supervised learning by classification**;

KEYWORDS

Entity typing; Mention typing; Multi-instance learning; Knowledge graph

ACM Reference Format:
Bo Xu, Zheng Luo, Luyang Huang, Bin Liang, Yanghua Xiao, Deqing Yang, Wei Wang. 2018. METIC: Multi-Instance Entity Typing from Corpus. In The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271804 .

*Yanghua Xiao is corresponding author. This paper was supported by National Key R&D Program of China under No.2017YFC1201200, by National Key Basic Research Program of China under No.2015CB358800, by National Key R&D Program of China under No.2017YFC0803700, by the National NSFC(No.61472085, No.U1509213, No.U1636207), by Shanghai Municipal Science and Technology Commission foundation key project under No.15JC1400900, by Shanghai Municipal Science and Technology project(No.16511102102, No.16JC1420401), Shanghai STCSM's R&D Program under Grant(No.16JC1420400)

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271804

1 INTRODUCTION

There has been a great amount of effort in trying to harvest knowledge from the Web in recent years, and a variety of *knowledge graphs* (KGs) or *knowledge bases* (KBs) have been constructed, such as Yago [22], DBpedia [1] and Freebase [2]. These knowledge bases play an important role in enabling machines to understand text or language. For example, without knowledge bases, the machines can only treat the sentence "Cristiano Ronaldo has won the 2016 Ballon d'Or award" as a string. Whereas, in contrast, knowledge bases enable the machine to recognize "Cristiano Ronaldo" as a famous Portuguese football player and the "Ballon d'Or award" as one of the most influential football awards. In this way, the machine *understands* that a player named Cristiano Ronaldo has won a football award called the Ballon d'Or award.

One of the fundamental components in KBs is *types* (or concepts) of entities. For example, Cristiano Ronaldo is classified as having the set of following types: {Person, Athlete, SoccerPlayer}. Characterizing an entity with different types is essential for many real applications such as relation extraction [9], recommender systems [15], question answering [6], emerging entities discovering [14], named entity disambiguation [30] and so on. Despite its importance, these types for entities are still far from being complete. For example, the average number of types in DBpedia is only 2.9 (5,044,223 entities with 14,760,728 types) [1], with many types still missing. Take Tom Cruise, who is a famous American actor and producer, as an example. In DBpedia, he is only listed under the type "Person", whereas he could be better described by a multitude of other types: "Actor", "Producer", etc.

Typing entities in a KB therefore becomes a great concern when building a knowledge base. In this paper, we focus on the KBs similar to DBpedia. More specifically, our aim is to enrich the types for entities that currently exist in CN-DBpedia [26], a popular Chinese KB. We will then review previous entity typing solutions based on structured features which still have some weaknesses in one aspect or another. This has motivated us to type entities from text corpus in CN-DBpedia.

[1]http://downloads.dbpedia.org/2016-10/statistics/, Retrieved Feb. 10, 2018.

1.1 Entity Typing by Structured Features

Existing systems [16, 19, 27] cast entity typing as a single-instance multi-label classification problem. For each entity e, they represent the entity with a set of features f_e extracted from KBs, and then infer the types of entity e from its feature set f_e. To make the solutions general enough, they often use the structured features of a KB (such as *attributes*, *attribute-value pairs* and *tags*) instead of the features extracted from free text such as part-of-speech tags, dependency parsing results, etc. [16]. As a result, their effectiveness is largely dependent on the quality of structured features. Unfortunately, the structured features are far from being perfect:

Firstly, *many entities don't have or have limited structured features, which disables the type inference from structured features.* Current KBs (such as DBpedia [10], Yago [22] and Freebase [2]) are mainly crafted from online encyclopedia websites, such as Wikipedia. According to Catriple [12], only 44.2% of articles in Wikipedia contain infobox information. Also in Baidu Baike [2], the largest Chinese online encyclopedia website, almost 32% (nearly 3 million) of entities lack the infobox and category information altogether [27].

Secondly, *entities may belong to multiple types, but their structured features only show the most typical one, which disables the inference of a complete set of types.* For example, from the structured features in the infobox on the right-hand side of Figure 1, which is a snippet of Donald Trump's Wikipedia page, we can only infer that Donald Trump belongs to type "President". We have no way to infer the other types such as "businessman" or "writer" are missing.

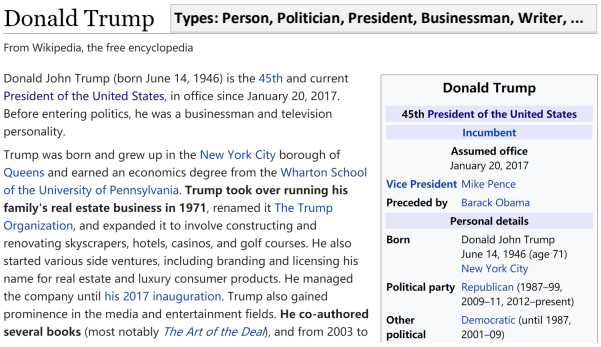

Figure 1: Information about Donald Trump. The left-hand and right-hand side are the abstract and infobox information in Wikipedia, respectively, and the upper side is the existing type information in KBs.

1.2 Entity Typing from Corpus

In this paper, we argue that structured features are not enough for entity typing, especially for the *complete* inference of *specific* types. We notice that an entity in knowledge bases contains both *structured* and *unstructured* features (such as description texts). Moreover, in most cases, description texts contain a richer set of information than structured features. For example in the left-hand side of Figure 1, which is Donald Trump's abstract text, we can infer from the sentence: "Trump took over running his family's real estate business in 1971", that Donald Trump also belongs to the type "businessman". We can also infer from the sentence: "He co-authored several books", that he could also be associated with the type "writer".

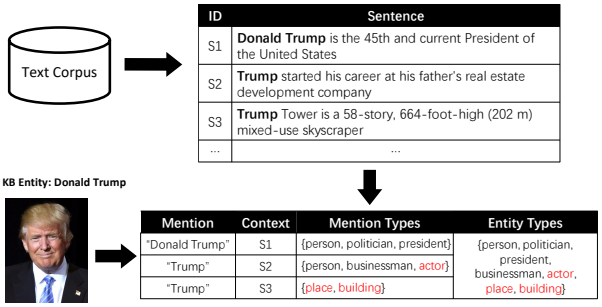

Figure 2: A Naïve method. Red labels within braces are errors.

In this paper, we focus on using the text of an entity to infer its types. The advantages for this approach are twofold. Firstly, *it is orthogonal to the structured features based typing solutions, and it can be used to complement existing solutions.* When an entity has sparse or less specific structured features, a corpus-based solution is the only real alternative. Even for entities that have rich structured features, the results of our solution could help complete and enrich the types found by structured features. Secondly, *the features from the corpus, or results derived from the corpus, could be used in an aggregation model (such as multi-view learning [23]) to achieve a better result.* By integrating the information in both structured and unstructured parts of the KB, it is more likely that we can achieve a better result for entity typing.

Many solutions are available for mention typing in sentences, we will first review this Naïve solution, which unfortunately still has weakness, and then propose our own solution.

1.2.1 Naïve Solutions. There is a significant difference between our problem setting and mention typing: namely that *an entity in KB usually has multiple mentions in texts while mention typing usually focusses on the typing for a single mention.* A very conservative estimate from in-text links on Wikipedia pages on the latest version of DBpedia [3] is that there are more than 5 million entities that have at least 2 mentions in the Wikipedia corpus. Therefore, to reuse existing mention typing solutions, we still need an appropriate strategy to aggregate the results derived from different mentions. The Naïve solution is therefore a simple union of the types found for each mention. We illustrate this procedure in Example 1.1. However, the Naïve method still has limited performance due to the following reasons:

Firstly, *mention typing is not a trivial problem and the current mention typing solutions still have a limited performance.* There are currently two categories of mention typing solutions. The first one relies on hand-crafted features [11, 19, 30] which relies on a great deal of human effort. Examples of this include: *Head*, *POS*, *Word*

[2] http://baike.baidu.com/

[3] http://downloads.dbpedia.org/2016-10/core-i18n/en/nif_text_links_en.ttl.bz2

Shape, *Length*, *Brown Cluster*, etc. Moreover, these features are usually dependent on the languages. The second one employs end-to-end neural network models, alleviating the difficulties of feature engineering [7]. However, despite these solutions, the performances of neural network models are still limited. This is illustrated by [19], which reported that neural network model HNM [7] didn't perform as well as many structured features based methods, such as FIGER [11], HYENA [30] and AFET [7], on public datasets including BBN, OntoNotes and Wiki.

Secondly, *the Naïve solution types each individual mention independently, ignoring the useful information that is gleaned in the collective typing of multiple mentions.* To use mention typing for entity typing, we first need to link mentions to appropriate entities and then type the entity. However, these two major steps are error-prone. For example, Trump in S3 is not the mention of the entity "Donald Trump", which is an error of entity mention discovery. Continuing along this example, we see that Trump in S2 also gets a wrongly predicted as an "actor", which is another error of mention typing. Many of these errors can be eliminated if we consider collective typing for all mentions instead of typing each mention independently and then aggregating. For example, the type "person" and "place" are mutually exclusive, meaning that an entity cannot simultaneously belong to both, implying that one of them is wrong, and needs to be removed.

Example 1.1 (An example for Naïve solution). As shown in Figure 2, the "Donald Trump" entity is mentioned in sentences S1, S2 and S3. We first type each mention in three sentences, and obtain the following type sets {person, politician, president}, {person, businessman, actor} and {place, building}, respectively. We then combine these prediction results and take the complete type set {person, politician, president, businessman, actor, place, building} as the types of entity Donald Trump, even though some of them, such as {actor, place, building}, are incorrect.

1.2.2 Our Idea. The weakness of the Naïve solution motivated us to propose a novel solution which we called METIC (**M**ulti-instance **E**ntity **T**yp**I**ng from **C**orpus). Our solution consists of two major steps: *mention typing* and *type fusion*. In the mention typing step, we infer types for each individual mention based on a *BiLSTM* model. Then, in type fusion, we employ an integer linear programming method to aggregate the types derived from each individual mention under the constraints derived from knowledge bases.

1.3 Contribution and Organization

Our contributions can be summarized as follows:

- Firstly, we propose a multi-instance method to enrich types for entities in KBs similar to DBpedia. We take each mention of an entity in KBs as an instance of the entity, and learn the types of an entity from multiple instances.
- Secondly, we propose a new neural network model for mention typing without using any hand-crafted features. Our experimental result shows that the model outperforms the state-of-the-art methods.
- Thirdly, we propose an integer linear programming (ILP) method for type fusion to overcome the incorrect prediction type results from mention typing. Experimental results show the effectiveness of our method.

The rest of this paper is organized as follows. In Section 2, we formalize the problem and give an overview of our proposed system *METIC*. In Section 3, we describe our neural network model for mention typing, whereas in section 4, we introduce our type fusion method. In Section 5, we describe our experimental settings for evaluation and discuss the results. In Section 6, we review the related work. Finally, we conclude this work in Section 7.

2 PRELIMINARY

In this section, we first formulate our problem, and then introduce the framework of our system: *METIC*.

2.1 Problem Formulation

Our task is to find a set of types/concepts for each entity in knowledge bases from its text corpus. Let T be the set of types, E be the set of all entities and D be the set of text corpus of entities in knowledge bases. For each entity $e \in E$, our goal is to find a subset of $T_e \subset T$ from its text corpus $D_e \subset D$, where D_e is the abstract information about entity e (like the abstract text of entity Donald Trump in Figure 1). For entity e, there may be many multiple mentions in corpus D_e. Let m_k be the k-th mention of e discovered in D_e and let c_k define the context information of m_k, including the sentence containing m_k and its position in D_e. Our problem consists of two major subtasks:

Subtask 1: Mention Typing. Given a mention m_k and its context c_k, the goal of mention typing is to predict the probability that m_k belongs to each type $t_i \in T$. We use an N-dimension vector $\vec{P}_{m_k}$ to represent the probability of mention m_k: $\vec{P}_{m_k} = \{P(t_1|m_k), P(t_2|m_k), \ldots, P(t_N|m_k)\}$, where N is the total number of target types T, and each dimension $P(t_i|m_k)$ denotes a probability that mention m_k belongs to type $t_i \in T$.

Subtask 2: Type Fusion. Given an entity $e \in E$ and its mentions $M_e = \{m_1, m_2, \ldots, m_k\}$, where each mention has a probability vector $\vec{P}_{m_k}$, the goal is to find a subset of $T_e \subset T$ for each entity e from the probability vectors: $\{\vec{P}_{m_1}, \vec{P}_{m_2}, \ldots, \vec{P}_{m_k}\}$.

2.2 Solution Framework

The framework of our solution is shown in Figure 3. There are two major phases in our solution: *offline* and *online*. In the offline phase, we train models for the two subtasks *mention typing* and *type fusion* separately. For mention typing, we model it as a multi-label classification. In our setting, we have both structured information and unstructured information for each entity, motivating us to use distant supervision method to construct the training data automatically and build a supervised learning model (more specifically we propose a neural network model). For type fusion, we model it as a constrained optimization problem, and propose an integer linear programming model in order to solve the problem. The constraints are derived from the semantic relationship between types.

In the online phase, we use the models built in the offline phase to enrich types for each entity in the KB. For each entity e, we first employ the existing entity linking systems, such as DBpedia spotlight [13] to discover entity mentions from D_e. Each mention and its corresponding context: $< m_i, c_i >$ are fed into the mention typing model. The model then derives a set of types with each type

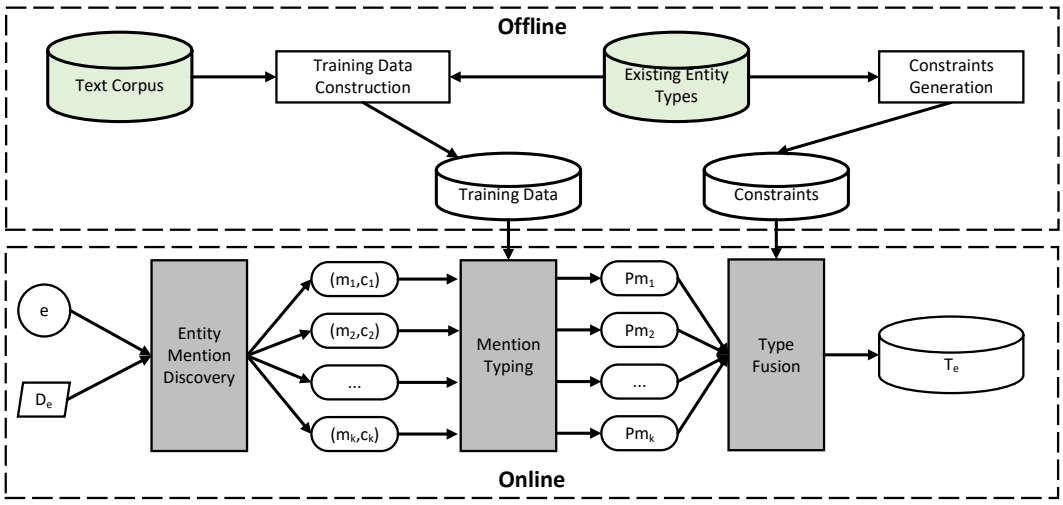

Figure 3: Framework of *METIC*.

being associated with a probability (i.e., $P(t|m_i)$). Types as well as their probability (i.e., $\vec{P}_{m_i}$) derived from each mention are further fed into the integer linear programming model with constraints specified as exclusive or compatibility among types. The model finally selects a subset from all candidate types as the final output types. We illustrate the procedure in Example 2.1

Table 1: The prediction results of our method. To save space, we only display those types of mentions with probability values greater than 0.5.

Mention	Context	Mention Types	Entity Types
"Donald Trump"	(S1, 0, 2)	{Person:0.9, Politician:0.9, President:0.85}	Person, Politician, President, Businessman
"Trump"	(S2, 0, 1)	Person:0.8, Businessman:0.7	
"Trump"	(S3, 0, 1)	Place:0.8, Building:0.7	

Example 2.1 (An example for our solution). Looking back on Example 1.1, the prediction results of our method shown in Table 1, we discover that for entity Donald Trump, there are three mentions in three sentences and we mark their positions. Then we use the mention typing method to get the prediction probabilities for each mention. Finally, we use our type fusion method to filter the errors and get the final typing results for the Donald Trump entity.

3 MENTION TYPING

Recall that we build a multiple-label classifier for mention typing. The classifier needs large amount of training data for the supervision. In this section, we first introduce how to construct the training data for mention typing, and then introduce our neural network model.

3.1 Training Data Construction

The state-of-the-art methods [7, 11, 30, 31] use distant supervision mechanism to automatically construct the training data. In our setting, many entities already have types in KBs. For example, as shown in Figure 1, entity "Donald Trump" has several types, such as "Person", "Politician", "President", "Businessman" and "Writer". These existing types allow us to use distant supervision to generate labeled data for mention typing. The process runs as follows: for each entity, which already has types in KBs, we first divide its abstract text into sentences. We then discover entity mentions from those sentences by existing entity linking systems, such as DBpedia spotlight [13]. Finally we assign the types of entities to their mentions in sentences.

Distant supervision, however, might produce *noise*, since it cannot take a mention's local contexts into account [20]. This noise data decreases the accuracy of the classifier. For example, in Figure 2, if we assign all of the types of entity "Donald Trump" to its mentions, then clearly many of the types such as "businessman" and "writer" are not as appropriate type labels as the "President" label inferred from S1. If we associate the wrong types with the mention, then the model will be misled in learning the wrong features. In general, *the type set of a mention is only a subset of the entity's complete type set.* We need an appropriate strategy to filter out the noisy types.

In this paper, we use a set of different mention typing models to create classifiers that serve as noise filters for the training data. This idea was first proposed by [3], then we apply it to mention typing problem. In order to not remove too much correct data, we only filter out the data that all of the noise filters consider to be noise. The procedure is illustrated in Example 3.1. There are four major steps:

- Firstly, we divide the training data into n parts.
- Secondly, for each part (say i-th part), we train m different classifiers that serve as noise filters on the other $n-1$ parts.
- Thirdly, we use each classifier to type each mention in the i-th part. For each classifier, we derive a *noisy set* consisting of types obtained from distant supervision but not predicted by the classifier. Each classifier believes that the types in the noisy set are mislabeled for the mention.

- Finally, when all of the m classifiers consider a certain type to be mislabeled, then we treat it as a wrong type. Finally, we remove all wrong types for mentions.

Table 2: The types of a mention obtained from distant supervision method are {A, B, C, D}. Each mention classifier produces a noisy set for the mention.

Classifier	Predicted Types	noisy set
1	{A, B, C}	{**D**}
2	{A, B}	{C, **D**}
3	{A, B}	{C, **D**}

Example 3.1 (Example for Noise Filtering). Suppose the type set of a mention obtained by distant supervision method is {A, B, C, D} and three classifiers are built. The noisy set for each classifier is shown in Table 2. The predicted mention types are {A, B, C}, {A, B} and {A, B}, respectively. Thus the noisy sets are therefore {D}, {C, D} and {C, D} , respectively. Since all the classifiers consider {D} to be the noise, we just remove it and keep {A, B, C} as the final types.

3.2 Neural Network Model

We propose a neural network model for mention typing. Compare with traditional works, such as Figer [11], HYENA [30] and AFET [19], we do not need any hand-crafted features and can apply for different languages.

We first pre-process the sentences. Similar to [7], in order to represent a sentence with a mention, we divide the sentence into three parts: the left context of the mention, the mention part and the right context of the mention, where the format is $[c_{-S} \cdots c_{-1}]$ $[m_1 \cdots m_N]$ $[c_1 \cdots c_S]$, and every part of them is length-fixed. S is the size of contexts, and N is the size of mention. c_{-i} represents the i-th word in the left context far from the mention, c_i represents the i-th word in the right context far from the mention, and m_i is the i-th word of the mention. The paddings are used to represent the absent words. An example is illustrated in Table 3.

Table 3: An example for Sentence Representation, where $S = 10$ and $N = 5$.

Format	$[c_{-S} \cdots c_{-1}]$ $[m_1 \cdots m_N]$ $[c_1 \cdots c_S]$
Sentence	**Donald Trump** is the 45th and current President of the United States
Left Context	[*Padding*, *Padding*, *Padding*, *Padding*, *Padding*, *Padding*, *Padding*, *Padding*, *Padding*, *Padding*]
Mention	[**Donald**, **Trump**, *Padding Padding Padding*]
Right Context	[is, the, 45th, and, current, President, of, the, United, States]

Our neural network model is shown in Figure 4. Each part of the sentence is fed into a parallel neural network with similar structure: a word embedding layer and a BiLSTM (bidirectional long short-term memory) layer [8]. We then concatenate the outputs of these BiLSTM layers to generate the final output.

3.2.1 Embedding layer. We use both word and character embedding to represent a word. For word embedding, we use the pre-trained word vectors from GloVe [17], which is trained on Wikipedia 2014 and Gigaword 5. Each word is represented by a 100-dimension vector.

To learn some sub-word patterns, such as morphemes (e.g., suffixes or prefixes) and roots, we also use character-based word embedding. Each word can be seen as a character sequence. We take the character sequence as input, and use the BiLSTM model to learn the 30-dimension vectors.

Finally, we use a 130-dimension vector to represent each word, which is the concatenation of 100-dimension word embedding and 30-dimension character-based word embedding.

3.2.2 BiLSTM layer. We then use the bi-directional LSTM layer to extract features from embeddings layer. Each of them is standard LSTM, and the memory cell for the t-th feature has the following composite functions:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ j_t &= \sigma(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= tanh(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= c_{t-1} \odot f_t + i_t \odot j_t \\ h_t &= tanh(c_t) \odot o_t \end{aligned} \tag{1}$$

where σ is a sigmoid function, and i, f, o, c are the input gate, forget gate, output gate and cell activation vectors, respectively. j is for computing new c value, and W and b are parameters for the LSTM layer.

3.2.3 Concatenation Layer. Finally, we concatenate entity mention and its context of the BiLSTM layers and fed into a dense classifier to predict the type of entity mention:

$$\sigma(W_o[h^1, h^2, h^3] + b_o) \tag{2}$$

where W_o and b_o are the parameters in this layer, and h^i is the i-th part of the model.

3.2.4 Model Training. We use *Keras* [4], which is a high-level neural networks library written in Python, to implement the model, and we use *binary cross-entropy* as our loss function [21]:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[y_{ij} \times log\hat{y_{ij}} + (1 - y_{ij}) \times log(1 - \hat{y_{ij}})\right]$$

where N is the number of sentence vectors, M is the number of types, y_{ij} is a binary indicator of whether or not type j is the correct classification for sentence vector i, and $\hat{y_{ij}}$ is the model probability of assigning type j to sentence vector i.

4 TYPE FUSION

As we mentioned in Example 1.1, aggregating all the types derived from mentions of an entity would produce noise. In this paper, we model type fusion as a constrained optimization problem, and propose an integer linear programming model (ILP) to solve it. *ILP* is

[4] https://keras.io/

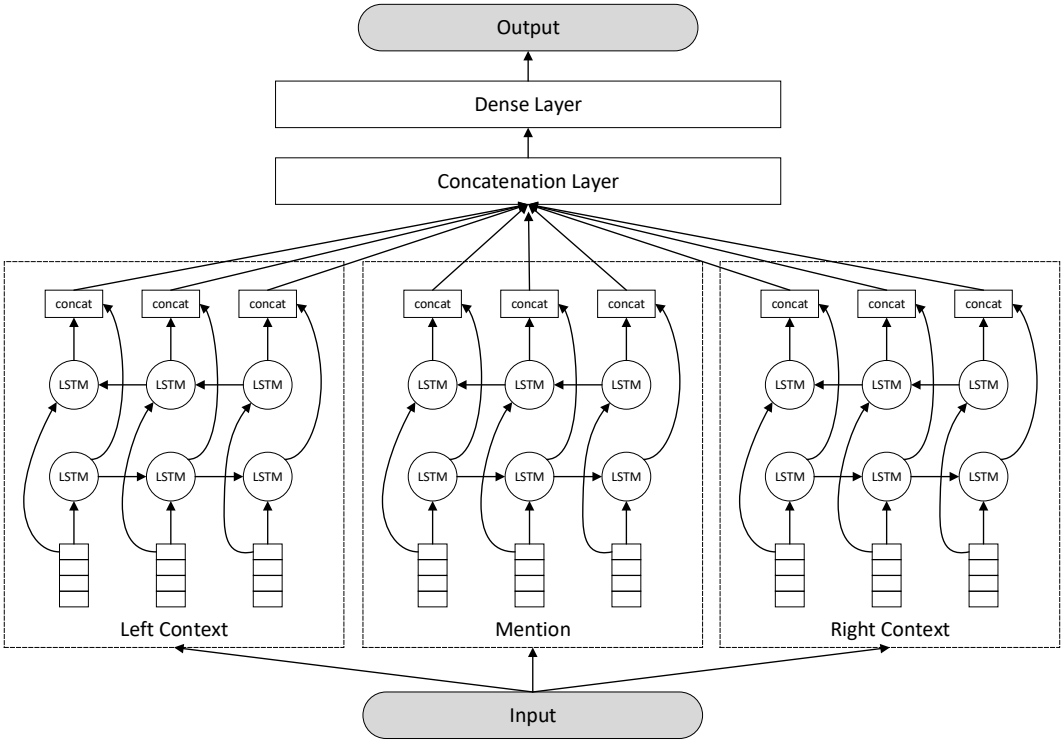

Figure 4: A Neural Network Model for Mention Typing.

an optimization model with constraints and all the variables are required to be non-negative integers [5].

4.1 Objective Function

For each entity e, we first define a decision variable $x_{e,t}$ for every candidate type t. These variables are binary and indicate whether entity e belongs to type t or not.

$$x_{e,t} = \begin{cases} 1 & , \text{ if entity } e \text{ belongs to type } t \\ 0 & , \text{ else} \end{cases}$$

We then define the objective function as follows:

$$max \sum_{t \in T} (\max_{m \in M_e} P(t|m) - \theta) \times x_{e,t} \tag{3}$$

, where $\max_{m \in M_e} P(t|m)$ represents the maximum probability that one mention of the entity e belongs to type t, and θ is the threshold (In our experiment, we set the threshold as 0.5).

4.2 Constraints

Then, we propose two constraints for our *ILP* model: a *type disjointness constraint* and a *type hierarchy constraint*. The purpose for this is twofold. First, we need to avoid trivial solutions. Notice that Equation 3 has a trivial solution in which all the binary variables are set to 1. Second, the accuracy can be further improved by incorporating prior knowledge about the semantic relationships between types.

(1) Type Disjointness Constraint. The *Type disjointness constraint* claims that *an entity can not belong to two semantically mutually exclusive types simultaneously*, such as *Person* and *Organization*. Types with no overlap in entities or an insignificant overlap below a specified threshold are considered to be *disjoint* [14].

(2) Type Hierarchy Constraint. The *Type hierarchy constraint* claims that if an entity does not belong to a type t then it will certainly not belong to any of t's sub-types. For example, an entity that does not belong to type Artist should not be classified to type Actor.

4.2.1 ***Constraint Generation.*** Specifically, to generate the type disjointness constraint, we use point-wise mutual information [4] (PMI) to measure the compatibility between types in existing entity typing results in KBs. As shown in Equation 4:

$$PMI(t_1, t_2) = \log \frac{P(t_1, t_2)}{P(t_1) \times P(t_2)} \tag{4}$$

, where $P(t_1)$ and $P(t_2)$ represent the probability that an entity belongs to type t_1 and t_2, respectively, and $P(t_1, t_2)$ represents the probability that an entity belongs to both type t_1 and t_2. The greater the value, the more compatible are these two types are with each other. Finally, we set a threshold α (In our experiment, we set the threshold as 0). If the PMI value of two types is less than α, then we consider these two types to be *mutually exclusive* (ME). We define $ME(t_1, t_2)$ to represent the mutually exclusive relationship between them. If two candidate types for an entity are disjoint,

the one with the lower probability of $\max_{m \in M_e} P(t|m)$ will be discarded.

To generate the type disjointness constraint, we use the *subClassOf* relations from KBs' ontologies. For example in DBpedia, type "Company" is a subclass of type "Organisation" and type "Businessman" is a subclass of type "Person". We define $IsA(t_1, t_2)$ to represent that type t_1 is a subclass of t_2.

4.3 ILP Model

With those two constraints above, we define our final *ILP* model as follows:

Maximize

$$\sum_{t \in T} (\max_{m \in M_e} P(t|m) - \theta) \times x_{e,t}$$

Subject to

$$\forall_{ME(t_1,t_2)} \quad x_{e,t_1} + x_{e,t_2} \leq 1$$
$$\forall_{IsA(t_1,t_2)} \quad x_{e,t_1} - x_{e,t_2} \leq 0$$
$$\forall_t \quad x_{e,t} \in \{0, 1\}$$

It is obvious that the first constraint restricts the selection of at most one of the two disjoint types will be selected. The second constraint restricts that when a super-type t_2 is not selected, every one of its sub-types will also be excluded. We use *PuLP*[5], which is an LP modeler written in python, to solve the problem. The aggregation procedure is illustrated in Example 4.1.

Table 4: A toy example for type fusion ($\theta = 0.5$). To save space, we have shortened Type "Person" and "Organisation" to "PER." and "ORG.", respectively.

	Agent	PER.	Athlete	Artist	Actor	ORG.
$P(t\|m_1)$	**0.9**	**0.9**	**0.8**	**0.3**	0.3	0.1
$P(t\|m_2)$	**0.9**	**0.9**	0.3	**0.3**	**0.6**	0.1
$P(t\|m_3)$	**0.9**	0.3	0.2	0.2	0.2	**0.6**
$\max P(t\|m)$	**0.9**	**0.9**	**0.8**	**0.3**	**0.6**	**0.6**
METIC-TF-N	1	1	1	0	1	1
METIC-TF-D	1	1	1	0	1	0
METIC-TF-H	1	1	1	0	0	1
METIC-TF	1	1	1	0	0	0

Example 4.1 (Entity Type Fusion). Let's suppose an entity e contains three mentions $M_e = \{m_1, m_2, m_3\}$, and that their probabilities $P(t|m)$ for type Agent, Person, Athlete, Artist, Actor and Organization are shown in Table 4. Then, when setting the θ to 0.5, their prediction type sets are {Agent, Person, Athlete}, {Agent, Person, Actor} and {Agent, Organization}, respectively. Therefore, we have that {Agent, Person, Athlete, Actor, Organization} is the candidate type set. Among those types, we can observe that Organisation type is disjointed with any other candidate types, and type Actor's super-type Artist is missing. Hence, by solely using type disjointness constraint, we can filter out the Organisation type. In addition to this, when we only use type hierarchy constraint only, we can filter out the Actor type. Using both of these constraints, we can simultaneously filter out the Organisation and Actor types together, enabling us to obtain accurate results.

[5]https://pypi.python.org/pypi/PuLP

5 EXPERIMENT

In this section, we present the experimental results. Specifically, we verify the effectiveness of mention typing and entity typing.

5.1 Data

5.1.1 Mention Typing. To evaluate the performance of our mention typing method, we use three public English datasets which have been widely used in mention typing task, namely BBN [24], OntoNotes [25] and Wiki [11]. We use the pre-processed data provided by [19] [6]. We notice that those datasets are totally different with the original ones reported by [19], implying that the evaluation results in [19] are invalid for those new datasets. The detailed differences shown in Table 5 are :

- **BBN dataset**. The BBN dataset contains 86,078 mentions from 32,739 sentences with 47 types for training, and 13,766 mentions from 6432 sentences with 48 types for testing. There are 9 types in the testing data that are not found in the training data.
- **OntoNotes dataset**. OntoNotes dataset contains 220,398 mentions from 88,284 sentences with 89 types for training, and 9,603 mentions from 1,312 sentences with 72 types for testing. There are 4 types in the testing data that were not found in the training data.
- **Wiki dataset**. Wiki dataset contains 2,690,286 mentions from 1,505,765 sentences with 128 types for training, and 563 mentions from 434 sentences with 50 types for testing.

Table 5: Statistics of the three public datasets, and the original ones reported by [19] also shown in parentheses.

Data Sets		Types	Sentences	Mentions
BBN	Train	47	32,739	86,078
	Test	48	6,432	13,766
	Total	53	39,171	99,844
	Total in [19]	(47)	(48,899)	(109,090)
OntoNotes	Train	89	88,284	220,398
	Test	72	1,312	9,603
	Total	93	89,596	230,001
	Total in [19]	(89)	(143,709)	(223,342)
Wiki	Train	128	1,505,765	2,690,286
	Test	50	434	563
	Total	128	1,506,199	2,690,849
	Total in [19]	(113)	(1.51M)	(2.69M)

5.1.2 Entity Typing. There are no public datasets to evaluate the performance of entity typing from corpus. The previous three datasets for mention typing can not be used to evaluate the performance of entity typing, since entities in those datasets only have one mention in their corpus. Hence, we constructed our own dataset as

[6]https://github.com/shanzhenren/AFET, Retrieved June 16, 2017.

follows: we use CN-DBpedia [26] as the knowledge base. CN-DBpedia provides both dump data [7] and APIs [8], which helped us to easily obtain Chinese entities' type information and text corpus. Finally, we obtained 1,741,119 mentions belonging to 55 types for training, and 33,006 entities along with 128,677 mentions for testing. Each entity contains at least 3 mentions in testing data.

5.2 Metrics

We use the widely used F1 metrics proposed by [11] to evaluate both the performance of mention typing and entity typing. Since their metrics are similar, in this section, we only present the metrics for entity typing. We denote the set of entities with ground truth types as T and the set of entities with valid predict results as P. For each entity e, we denote the ground truth types as t_e and the predict set as p_e. The metrics are listed as follows:

- **Strict Accuracy**:

$$precision = (\sum_{e \in P \cap T} \delta(t_e = p_e))/|P|$$

$$recall = (\sum_{e \in P \cap T} \delta(t_e = p_e))/|T|$$

- **Loose Macro**:

$$precision = \frac{1}{|P|} \sum_{e \in P} \frac{|p_e \cap t_e|}{p_e}$$

$$recall = \frac{1}{|T|} \sum_{e \in T} \frac{|p_e \cap t_e|}{t_e}$$

- **Loose Micro**:

$$precision = \frac{\sum_{e \in P} |t_e \cap p_e|}{\sum_{e \in P} |p_e|}$$

$$recall = \frac{\sum_{e \in T} |t_e \cap p_e|}{\sum_{e \in T} |t_e|}$$

We denote *Acc.*, *Ma-F1* and *Mi-F1* to represent the f1 scores for *Strict Accuracy*, *Loose Macro* and *Loose Micro*, respectively, where f1 scores are computed by corresponding *precision* and *recall* as follows:

$$f1 = \frac{2 \times precision \times recall}{precision + recall}$$

5.3 Performance of Mention Typing

5.3.1 Baselines. We compare our method and its variants with some state-of-the-art methods, including structured features based methods and neural network models. More specifically these are:

- **AFET** [19]. The state-of-the-art approach that use structured features.
- **HNM** [7]. The first neural model for mention typing.
- **HNM-ML**. A variant of **HNM** for multi-label problem.
- **METIC-MT**. Our neural network model for mention typing.
- **MT-SP**. A variant of our model that doesn't split a sentence into three parts, but instead uses a position flag, where 1 represents that it is a part of the mention and 0 represents that it does not.
- **METIC-MT-Filtered**. A variant of our model that has filtered training data. In this experiment, we use HNM-ML, METIC-MT and MT-SP as noise filters to clean the training data.

5.3.2 Performance Comparisons and Analyses. The comparison results are shown in Table 6. Compared to the structured features based method, namely AFET, METIC-MT performs better on Wiki and BBN datasets, while AFET performs a little better on OntoNotes, which demonstrates the predictive power of the neural network models without using any hand-crafted features. Compared to other neural network models, namely HNM, HNM-ML and MT-SP, METIC-MT also performs better, demonstrating that a well-designed neural network model can achieve a better result. METIC-MT-Filtered outperforms METIC-MT, demonstrating the effectiveness of our noise filtering strategy.

5.4 Performance of Entity Typing

5.4.1 Baselines. We compared the performances of different entity typing strategies by combining different mention typing and type fusion methods. We use HNM-ML, METIC-MT and MT-SP as the mention typing methods and define the different type fusion methods as follows:

- **METIC-TF**. Our ILP model that contains both of two constraints.
- **METIC-TF-N**. A variant of our model with no(**N**) constraints.
- **METIC-TF-D**. A variant of our model that only contains the type disjointness(**D**) constraint.
- **METIC-TF-H**. A variant of our model that only contains the type hierarchy(**H**) constraint.

5.4.2 Performance Comparisons and Analyses. Using the same type fusion method, different mention typing methods would achieve different results. From Table 7, we see that METIC-MT method also achieve the best performance, which is consistent with the previous results on the three public datasets. We also observe that by using the same mention typing method, different type fusion methods would also achieve different results. Our type fusion method METIC-TF that contains both of the two constraints performs better than the others, demonstrating that both of the two constraints are effective. Comparing METIC-TF-D with METIC-TF-H, METIC-TF-D achieves a better result than METIC-TF-H, demonstrating that the constraint of *type disjointness* works better than *type hierarchy*.

5.5 Case Study

Table 8 shows the case study of entity typing on the Chinese entity "The south of Fujian Province". We use the same mention typing method (METIC-MT) along with different type fusion methods. Compared to other methods, our approach METIC-TF can avoid non-hierarchical predictions (e.g., "Infrastructure") and disjoint predictions (e.g., "Agent" and "Person").

6 RELATED WORK

In this section, we review and summarize the works that are most relevant to our research. These include works in mention typing, entity typing and label noise reduction.

[7] http://kw.fudan.edu.cn/cndbpedia/download/
[8] http://kw.fudan.edu.cn/apis/cndbpedia/

Table 6: Performance of mention typing on the three public datasets.

Mention Typing Method	Wiki			OntoNotes			BBN		
	Acc.	Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1
AFET	0.205	0.616	0.600	**0.348**	**0.620**	**0.559**	**0.638**	0.698	0.710
HNM	0.442	0.670	0.616	0.344	0.527	0.479	0.216	0.558	0.521
HNM-ML	0.471	0.657	0.683	0.267	0.503	0.493	0.469	0.682	0.680
MT-SP	0.469	0.661	0.680	0.337	0.556	0.517	0.497	0.708	0.702
METIC-MT	0.528	0.711	0.718	0.256	0.524	0.508	0.492	0.717	0.710
METIC-MT-Filtered	**0.551**	**0.745**	**0.735**	0.313	0.571	0.539	0.520	**0.718**	**0.713**

Table 7: Performance of different entity typing strategies ($\theta = 0.5$).

Mention Typing	Type Fusion	Acc.	Ma-F1	Mi-F1
HNM-ML	METIC-TF-N	0.46	0.744	0.739
	METIC-TF-H	0.461	0.745	0.74
	METIC-TF-D	0.635	0.749	0.781
	METIC-TF	0.636	0.749	0.782
MT-SP	METIC-TF-N	0.648	0.879	0.862
	METIC-TF-H	0.651	0.88	0.863
	METIC-TF-D	0.791	0.893	0.906
	METIC-TF	0.792	0.894	0.906
METIC-MT	METIC-TF-N	0.659	0.905	0.865
	METIC-TF-H	0.662	0.905	0.865
	METIC-TF-D	0.798	0.907	0.911
	METIC-TF	**0.799**	**0.907**	**0.911**

Table 8: Case study of entity typing on the entity "The south of Fujian Province**". Using the same mention typing method (**METIC-MT**) along with different type fusion methods. The bold ones are the correct types.**

Strategy	Prediction Type Set
METIC-TF-N	**Place**, **PopulatedPlace**, **Settlement**, Agent, Person, Infrastructure
METIC-TF-H	**Place**, **PopulatedPlace**, **Settlement**, Agent, Person
METIC-TF-D	**Place**, **PopulatedPlace**, **Settlement**, Infrastructure
METIC-TF	**Place**, **PopulatedPlace**, **Settlement**
Ground Truth	**Place**, **PopulatedPlace**, **Settlement**

Mention typing is a task that classifies an entity mention in a sentence into a predefined set of types [18]. Recent works have focused on a large set of fine-grained types [7, 11, 30, 31], since their types are not mutually exclusive, they cast the problem as multi-label classification problems. However, most of them use hand-crafted features, such as *Head*, *POS*, *Word Shape*, *Length*, *Brown Cluster*, etc, which requires great human effort. [7] proposes the first neural network model for mention typing, but only suits for single-label classification. In this paper, we extend this work to apply for multi-label classification.

Entity typing is a task that classifies an entity in knowledge bases into a predefined set of types. An entity in a knowledge base includes a lot of rich information, such as structure (infobox, category, etc.) and text features. Great efforts have been devoted to using structured features to infer their types [16, 22, 27]. However, the effectiveness is largely dependent on the completeness of structure features, which unfortunately is not guaranteed in real data set. Hence, in this paper, we propose to use the plain text description of an entity to infer its types. [28, 29] propose some neural network models to solve this problem, but they do not consider the constraints between types.

Label noise reduction problem in mention typing was first studied by [20], which is identifying the correct type labels for each mention from its noisy candidate type set. However, this is still problematic since extracting hand-crafted features from texts is also required. Therefore, inspired by [3], we have decided to use multiple classifiers to filter the noisy training data, and avoiding the need for any hand-crafted features.

7 CONCLUSION

In this paper, we propose a multi-instance entity typing system *METIC* to type entities in knowledge bases from text corpus. We first construct training data by using a distant supervision method. However, we discovered that the type labels obtained from the knowledge bases are usually noisy. We therefore used a noise filtering method to clean the noisy labels. For the mention typing step, we propose an effective neural network model to type each mention of an entity. Then for the type fusion step, we use an integer linear programming method to aggregate all the types derived from mentions of an entity. Our noise filtering strategy is based on two constraints, type disjointness and type hierarchy constraints. Experimental results show the effectiveness of our method.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The semantic web*. Springer, 722–735.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1247–1250.

[3] Carla E. Brodley and Mark A. Friedl. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research* 11 (1999), 131–167.

[4] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16, 1 (1990), 22–29.

[5] James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31 (2008), 399–429.

[6] Wanyun Cui, Yanghua Xiao, and Wei Wang. 2016. KBQA: An Online Template Based Question Answering System over Freebase. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*. 4240–4241.

[7] Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 1243–1249.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 541–550.

[10] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal* 5 (2014), 1–29.

[11] Xiao Ling and Daniel S Weld. 2012. Fine-Grained Entity Recognition.. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 94–100.

[12] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. 2008. Catriple: Extracting triples from wikipedia categories. In *Asian Semantic Web Conference*. Springer, 330–344.

[13] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*. ACM, 1–8.

[14] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained Semantic Typing of Emerging Entities.. In *ACL (1)*. 1488–1497.

[15] Alexandre Passant. 2010. dbrec–music recommendations using DBpedia. In *The Semantic Web–ISWC 2010*. Springer, 209–224.

[16] Heiko Paulheim and Christian Bizer. 2013. Type inference on noisy rdf data. In *International semantic web conference*. Springer, 510–525.

[17] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[18] Xiang Ren, Ahmed El-Kishky, Heng Ji, and Jiawei Han. 2016. Automatic Entity Recognition and Typing in Massive Text Data. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2235–2239.

[19] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*. 1369–1378.

[20] Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. 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*. ACM, 1825–1834.

[21] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[22] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

[23] Shiliang Sun. 2013. A survey of multi-view machine learning. *Neural Computing and Applications* 23, 7-8 (2013), 2031–2038.

[24] Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia* 112 (2005).

[25] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. OntoNotes Release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium* (2011).

[26] Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Yanghua Xiao. 2017. CN-DBpedia: A Never-Ending Chinese Knowledge Extraction System. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 428–438.

[27] Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. 2016. Cross-Lingual Type Inference. In *International Conference on Database Systems for Advanced Applications*. Springer, 447–462.

[28] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise Mitigation for Neural Entity Typing and Relation Extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1. 1183–1194.

[29] Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level Fine-grained Entity Typing Using Contextual Information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 715–725.

[30] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *24th International Conference on Computational Linguistics*. ACL, 1361–1370.

[31] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2013. HYENA-live: Fine-Grained Online Entity Type Classification from Natural-language Text.. In *ACL (Conference System Demonstrations)*. 133–138.