Knowledge-aware Named Entity Recognition with Alleviating Heterogeneity

Binling Nie,[1*] **Ruixue Ding,** [2] **Pengjun Xie,** [2] **Fei Huang,** [2] **Chen Qian,** [3] **Luo Si** [2]
[1] HangZhou Dianzi University
[2] Alibaba Group
[3] Tsinghua University
binlingnie@hdu.edu.cn, {ada.drx, f.huang, luo.si}@alibaba-inc.com,
chengchen.xpj@taobao.com, qc16@mails.tsinghua.edu.cn

Abstract

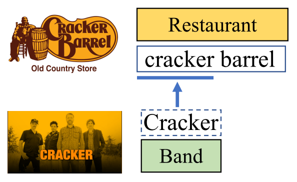
Named Entity Recognition (NER) is a fundamental and important research topic for many downstream NLP tasks, aiming at detecting and classifying named entities (NEs) mentioned in unstructured text into pre-defined categories. Learning from labeled data only is far from enough when it comes to domain-specific or temporally-evolving entities (*e.g.* medical terminologies or restaurant names). Luckily, open-source Knowledge Bases (KBs) (*e.g.* Wikidata and Freebase) contain NEs that are manually labeled with predefined types in different domains, which is potentially beneficial to identify entity boundaries and recognize entity types more accurately. However, the type system of a domain-specific NER task is typically independent of that of current KBs and thus exhibits heterogeneity issue inevitably, which makes matching between the original NER and KB types (*e.g.* `Person` in NER potentially matches `President` in KBs) less likely, or introduces unintended noises without considering domain-specific knowledge (*e.g.* `Band` in NER should be mapped to `Out_of_Entity_Types` in the restaurant-related task). To better incorporate and denoise the abundant knowledge in KBs, we propose a new KB-aware NER framework (KaNa), which utilizes type-heterogeneous knowledge to improve NER. Specifically, for an entity mention along with a set of candidate entities that are linked from KBs, KaNa first uses a *type projection* mechanism that maps the mention type and entity types into a shared space to homogenize the heterogeneous entity types. Then, based on projected types, a noise detector filters out certain less-confident candidate entities in an unsupervised manner. Finally, the filtered mention-entity pairs are injected into a NER model as a graph to predict answers. The experimental results demonstrate KaNa's state-of-the-art performance on five public benchmark datasets from different domain.

Introduction

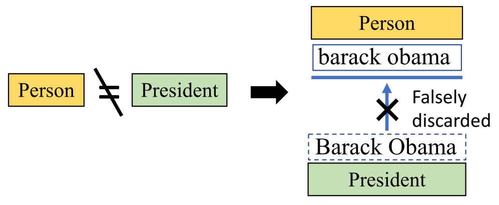
Named Entity Recognition (NER) is a fundamental and important research topic in the area of natural language processing (NLP), aiming at detecting and classifying *named entities* (NEs) mentioned in unstructured text into predefined categories (Ngo, Dien, and Winiwarter 2014). The data-driven methods, *e.g.* conditional random fields (CRF) (Li, Bontcheva, and Cunningham 2005) and BiLSTM-CRF (Lample et al. 2016), are widely used for NER due to their power to effectively capture the sequential dependency observed in training data. Nevertheless, simply learning from the labeled data is far from enough when it comes to domain-specific or temporally-evolving entities (*e.g.* medical terminologies or restaurant names). Luckily, many open-source Knowledge Bases (KBs) (*e.g.* Wikipedia and Freebase) (Vrandečić and Krötzsch 2014) contain the well-structured NEs that are manually assigned with predefined types in different domains, which is potentially beneficial for NER to locate entity boundaries and recognize entity types. For example, in the text "*colon carcinoma cells*", utilizing the prior knowledge that the mention "*colon carcinoma*" is associated with the KB entity *colorectal cancer* with the type of `Disease`, it generally falls into the `Disease` category with a much higher probability than other candidates such as *colon* or *carcinoma*.

However, the key challenge of incorporating KB knowledge into NER task is the *heterogeneity* issue between KB and NER type systems. Specifically, there are usually thousands of types in a KB type system but a NER type system usually has no more than hundred of types (even less), which shows a gap and further leads to a divergent design philosophy. We here summarize the main issues derived from the heterogeneity problem into the following aspects: 1) Type system generalizability. KB types are predefined relatively stably for general purposes while NER types are domain-specific (*i.e.* varying from one task to another). 2) Different namespace. The possible names of the same entity type could be largely different among different type systems. 3) One-to-many matching. A NER entity usually has one type but a KB entity often has multiple types. The three classes of heterogeneity would cause the following problems respectively if careful designs were not considered: 1) Task-irrelevant noise. Since the number of entity types in NER is much smaller than KB types and NER types are usually task specific, many task-irrelevant entities could be unintentionally introduced via the use of KBs. Take a restaurant-related NER task as an example, as shown in Figure 1 (a), the subword *cracker* in "*cracker barrel*" (a restaurant name) is linked to a KB entity *Cracker* with type `Band`. The association between mention *cracker* and KB entity *Cracker* could mislead model to predict wrong entity boundaries and types. 2) Complex Noise Identification.

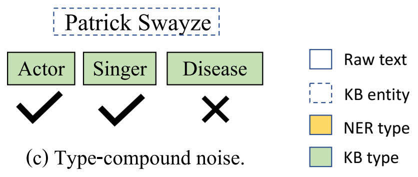
[*]Corresponding author



(a) Task-irrelevant noise.



(b) Complex Noise Identification.



(c) Type-compound noise.

Figure 1: Problems caused by the type heterogeneity.

Different namespace increases the difficulties of denoising. A straightforward method to filter out task-irrelevant noise is to only preserve those KB entity types in the intersection of two type systems. As a result, those entities which are semantically-helpful but morphologically-different could be largely ignored. As shown in 1 (b), the link from text *barack obama* with type `Person` to entity *Barack Obama* with type `President` is not noise but it is discarded since `President` and `Person` are morphologically-different. These also could be resolved by manually defining rules at the cost of manual costs and low generalizability: all human efforts need redoing once facing a new NER task. 3) Type-compound noise. As shown in Figure 1 (c), a person entity in KB could have the `Disease` type when he/she was dead of this but NER normally not. Thus, the types of one KB entity should be considered as a whole, otherwise they could probably be another noise in NER. To address issues below, (He et al. 2020) incorporated KB knowledge in NER by a multi-task pretraining of KB entity embeddings, where KB type knowledge was discarded so the task-irrelevant noise was introduced. (Rijhwani et al. 2020) inserted the KB type features via an auto-encoder, but the incompatibility between two type systems was resolved at the expense of expensive manual costs.

To better incorporate and denoise the abundant knowledge in KBs, we propose a new KB-aware NER framework (KaNa), which utilizes type-heterogeneous knowledge to improve NER. Our method consists of three main modules: *knowledge denoising*, *knowledge injection* and *knowledge-based inference*. The knowledge denoising module serves to select task-related, type-compound coherent entities retrieved from KBs. This is realized by type projectors which can project a mention type and some KB entity types into a common space and a noise detector which filters out entities according to their projected types. The training of knowledge denoising module is in an unsupervised[1] manner, without the need of extra human labors. Knowledge injection builds raw text and its relations with selected KB knowledge in form of a graph. After encoding the graph by Graph Attention Networks (GATs) (Veličković et al. 2018), nodes representing raw text are aware of selected knowledge. Finally, the node representations of raw text are fed to a standard CRF layer to predict answers.

To summarize, we make the following main contributions:

- We propose a novel knowledge-aware NER framework, which can effectively incorporate KB knowledge into NER tasks by alleviating the unintended heterogeneity issue. To the best of our knowledge, this work is the first attempt to alleviate type-related knowledge discrepancy in KBs for NER without extra human efforts.
- To tackle the heterogeneity issue between NER and KB type systems, we propose the knowledge denoising mechanism. The training of the module is in an unsupervised manner, without the need of extra labeled data in the knowledge-injected processes.
- The experimental results on five datasets from different domains show that our method consistently achieves state-of-the-art performance, validating the effectiveness of our proposed method and mechanisms.

Related Work

Many supervised methods have been successively proposed for NER. Traditional statistical methods, like Hidden Markov Models (HMM), Super Vector Machines (SVM), CRF or decision trees, followed some traditional feature-engineering-based paradigm to fit data and make predictions (Li, Bontcheva, and Cunningham 2009; Passos, Kumar, and Mccallum 2014; Luo et al. 2015). However, this type of strategies relies on a set of handcrafted features and/or does not take into account the inherent dependencies across contexts. To remedy this, a large number of neural network architectures emerged to utilize different representations of characters, words, sub-word units or any combinations of these to assign proper categories to words in a sentence. (Strubell et al. 2017; Ma and Hovy 2016; Peters et al. 2017; Cui and Zhang 2019; Lu, Bai, and Langlais 2019; Chen et al. 2019; Huang, Xu, and Yu 2015; Lample et al. 2016; Tran, Mackinlay, and Yepes 2017). However, merely learning from finite labeled data is far from enough when it comes to domain-specific or temporally-evolving entities (*e.g.* medical terminologies or restaurant names), which creates a demand for extra knowledge injection.

Most previously, KBs are used to create NER training data (Kazama and Torisawa 2007; Richman and Schone

[1]Note that the mentioned "unsupervised" is only limited in the proposed modules, and the original NER task is still supervised.

2008; Nothman et al. 2013; Morgan et al. 2003; Vlachos and Gasperin 2006). A prominent technique is to follow links back from KB articles to documents that mention the subject of the article, heuristically labelling high-precision matches to create training data. This has been used for genetic KBs (Morgan et al. 2003; Vlachos and Gasperin 2006), and Wikipedia (Kazama and Torisawa 2007; Richman and Schone 2008; Nothman et al. 2013).These works do not consider our setting where gold-standard entities are given at training as their goal is to generate training data. Recently, He et al. (2020) pretrained KB entity embeddings for NER based on the entity relations defined in KB where the knowledge type was unused so the task-irrelevant noise was introduced. Radford, Carreras, and Henderson (2015) explored to improve NER performance by infusing text with KB knowledge under a *task specific setting*. In this setting, we have metadata for each document in the form of document-specific KB tags, which doesn't consider task-irrelevant noise and the difference of namespace. In real scenario, the incompatibility between two type systems is resolved at the expense of expensive manual costs. Rijhwani et al. (2020) proposed the SoftGaz model which adapted the BMEOW feature encoding mechanism by using an autoencoder for neural models, which is the state-of-the-art KB-aware NER model but only works under a task specific setting.

Methodology

Task Definition

We start by giving the formal definition of NER. Given an input text $X = \langle x_1, x_2, \cdots, x_n \rangle$ with length n, the task is to output a tag sequence $Y = \langle y_1, y_2, \cdots, y_n \rangle$, where $y_i \in \mathcal{Y}, 1 \leq i \leq n$ and $\mathcal{Y}$ is the set of predefined tags following the BIO tagging scheme. Each tag in $\mathcal{Y}$ is composed of the boundary of the entity and the entity type. Specifically, 'B-', 'I-', 'O' denote the beginning, inner, outside of an entity name respectively.

Model Overview

We propose a new KB-aware NER framework (KaNa) and the overall architecture of KaNa is shown in Figure 2. KaNa can be divided into three parts including: knowledge denoising, knowledge injection and inference. For every mention in a text, we retrieve candidate entities from KBs. Then, every mention-entity pair is fed to the knowledge denoising module, which makes discard-select decision by comparing the projected mention and entity types. Then, text infused with selected knowledge is modeled in form of a text-entity graph, which can be further encoded by GATs to obtain knowledge-aware context embeddings for every word. Finally, these knowledge-aware embeddings are fed to a CRF to predict output.

Type System Heterogeneity

In this section , we briefly describe the problem of type system heterogeneity informally. For those widely-used KBs (*e.g.* Wikidata), their type systems are quite different from NER type system. For a typical NER task, several types (*e.g.* `Disease`,`Restaurant`) are defined. The tag set varies from one task to another. There are few NER tasks with more than 100 types. Except for the nested NER task, one entity usually has only one type. However, for a KB type system, there are tens of thousands of types which form a tree structure. A KB entity can have tens of types associated to accurately describe it (*e.g. Buger_King* is associated with `popstra.company`, `dining.restaurant`, `common.topic`, `popstra.restaurant`, *etc.*).

Knowledge Denoising

For a mention $m \in \{x_i ... x_j, 1 \leq i \leq j \leq n\}$ in X, we generate related KB entities $\mathcal{C}(m) = \{e_1, ..., e_k\}$. For every mention-entity pair in $\mathcal{C}(m)$, due to the heterogeneity of two type systems, it could be noise even with prior being 100%. As the example case "*Patrick Swayze died of pancreatic cancer.*" in Figure 2, aside from the shown pair example, *Patrick Swayze* is also linked to an `Actor` entity, which should be filtered out since it is task-irrelevant. Therefore we put forward a denoising module, where a type projection mechanism is proposed to map the mention type (obtained via a FC layer) and entity types into a shared space to homogenize the heterogeneous entity types. Then, based on projected types, a noise detector filters out certain less-confident candidate entities in an unsupervised manner. The training of knowledge denoising module is finished before the training of knowledge injection and inference module.

Type System Projector For an entity $e \in \mathcal{C}(m)$ associated with $|T|$ types $T = \{t_{e,j}\}_{1 \leq j \leq |T|}$, we first represent each type by a d_k dimension vector $v(t_{e,j})$. Then, we sum over all type vectors to get its type embeddings. Later, a KB type projector P_{kb} is used to map the type embeddings into a common type space of dimension d_c:

$$\begin{aligned} r_e &= P_{kb}(\frac{1}{|T|} \sum_{j=1}^{|T|} v(t_{e,j})) \\ P_{kb}(x) &= ReLU(W_{kb} x + b_{kb}) \end{aligned} \tag{1}$$

where $W_{kb} \in \mathbb{R}^{d_c \times d_k}$ and $b_{kb} \in \mathbb{R}^{d_c}$ are learned parameters. r_e is the type representation of entity in common type space and it could be seen as a summary of original types, serving to reduce type-compound noise.

The mention embeddings are calculated by concatenating its surrounded LSTM hidden states. Specifically, every word x_i in the input text X is encoded by a look-up table $W_w \in \mathbb{R}^{d_w \times |V|}$:

$$w_i = W_w(x_i) \tag{2}$$

where $|V|$ is the size of vocabulary and d_w is the word embeddings dimension. Word vectors are fed to forward and backward LSTM models to get contextual information:

$$\begin{aligned} \overrightarrow{h_i} &= \overrightarrow{LSTM}(w_i, \overrightarrow{h_{i-1}}) \\ \overleftarrow{h_i} &= \overleftarrow{LSTM}(w_i, \overleftarrow{h_{i-1}}) \end{aligned} \tag{3}$$

The mention embeddings e_m of $m = x_{i:j}$ is formed of:

$$e_m = [\overrightarrow{h_{i-1}}, \overleftarrow{h_{i-1}}, \overrightarrow{h_{j+1}}, \overleftarrow{h_{j+1}}] \tag{4}$$

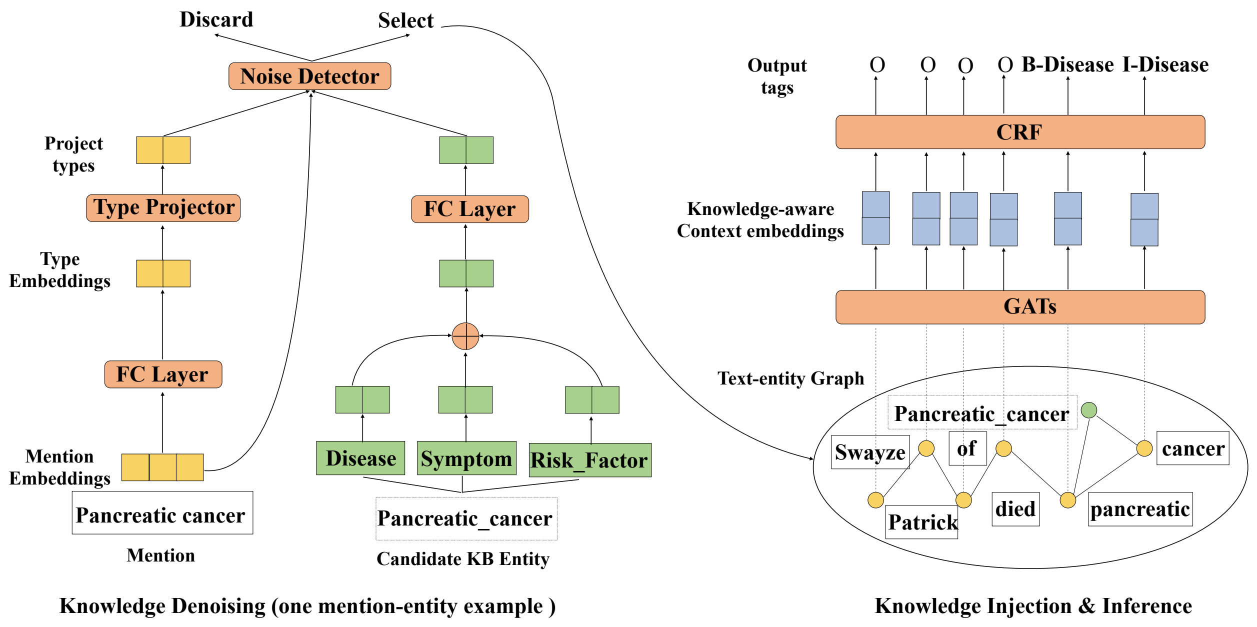


Figure 2: Model Architecture. Take the sentence "*Patrick Swayze died of pancreatic cancer.*" as an example. For an entity mention *Pancreatic cancer* along with a set of candidate entities (*e.g. Pancreatic_Cancer*) that are retrieved from KBs, KaNa first uses a type projection mechanism that maps the mention type (obtained via a FC layer) and entity types into a shared space to homogenize the heterogeneous entity types. Then, a noise detector filters out certain less-confident candidate entities in an unsupervised manner. Finally, the filtered mention-entity pairs are injected into a text-entity graph, in which GATs are used to encode the graph. Once obtaining those knowledge-aware node embeddings, KaNa feeds the graph-interacted context embeddings into a standard CRF layer for the final prediction.

We get the type embeddings t_m of m by a fully connected layer:

$$t_m = W_t e_m + b_t \tag{5}$$

where $W_t \in \mathbb{R}^{|L|\times 4d_h}$, $b_t \in \mathbb{R}^{|L|}$ are learned parameters, $|L|$ is the size of NER type set, d_h is the dimension of LSTM hidden states.

The mention type projector P_{ner} maps t_m into the common type space:

$$\begin{aligned} r_m &= P_{ner}(t_m) \\ P_{ner}(x) &= W_{ner}x + b_{ner} \end{aligned} \tag{6}$$

where $W_{ner} \in \mathbb{R}^{d_c\times|L|}$, $b_{ner} \in \mathbb{R}^{d_c}$ are learned parameters, r_m is the type representation of mention in common type space. From now on, both mention type and entity types are mapped to common type space which reduces the different namespace problem when comparing types between two type systems.

Noise Detector We get now type representations for both mention and entity in a shared space. The noise detector can calculate the probability of an entity being noise in a given context by comparing projected type embeddings:

$$f_{nd}(m,e) = P_N(0|m,e) = W_n([r_e, r_m, e_m]) + b_n \tag{7}$$

where $W_n \in \mathbb{R}^{1\times(4d_h+2d_c)}$ and $b_n \in \mathbb{R}$ are learned parameters, f_{nd} is the noise detector function.

Unsupervised Training We train type projectors along with noise detector in an unsupervised manner since the labeled data for noise detection is unavailable. The data of this task is in form of mention-entity pairs with labels showing whether they are useful knowledge or not.

The positive samples are hard to find. The type of mention m is given in NER training data. In this part, we use mentions not being tagged with 'O' to generate samples. For a mention m, entities in the candidate set $C(m)$ generally could have a higher probability being positive, but we have no access to gold entities. However, we know there could be at least one positive entity if KB was big enough. This setting is similar to the one of Multi Instance Learning (MIL) (Zhou and Zhang 2006). Instead of receiving a set of instances which are individually labeled, the learner of MIL receives a set of labeled bags, each containing many instances. Thus, we propose using the candidate mention set $\mathcal{C}(m)$ as positive samples bag $T_{pos} = \mathcal{C}(m)$.

We use negative sampling to better discriminate useful knowldge from noise. The negative samples are easy to construct. Since NER types are of limited size. So we can use those mentions which are not tagged with 'O' to generate negative samples by randomly selecting KB entities under obviously fault root types. For example, if the mention type is $Disease$, KB types under roots like $Vehicle$, $Company$ are obviously fault. We form the negative samples T_{neg} by this simple but effective way.

For every mention m, we now have negative samples T_{neg} and positive samples bag T_{pos}. We train our noise detector to score at least one candidate in T_{pos} higher than any candidate in T_{neg}. This is achieved by using max-margin loss. The loss of denoising task can be described as:

$$l_{nd}(m,e) = max(0, \max_{e \in T_{neg}} f_{nd}(m,e) - \max_{e \in T_{pos}} f_{nd}(m,e) + \delta)$$
$$L_{nd} = \sum_{m,e \in D} l_{nd}(m,e) \quad (8)$$

where δ is a margin, D is the automatically constructed training set. The objective is that the distance between positive and negative samples is greater than δ.

By training in such unsupervised manner, we get our trained knowledge denoising module which can then help denoise knowledge and send selected mention-entity pairs to knowledge injection and inference module.

Knowledge Injection

After knowledge denoising, we get some selected mention-entity pairs. We need to integrate such knowledge into NER model. Recently, graph based models (Gui et al. 2019; Sui et al. 2019; Ding et al. 2019) are widely used to incorporate external knowledge in NER. They model boundary and type information along with text explicitly in form of a graph. Then, knowledge-aware text features are calculated by a node propagation mechanism following works of graph representation learning (Perozzi, Alrfou, and Skiena 2014; Tang et al. 2015; Grover and Leskovec 2016; Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017). Results show explicit boundary and type information are more effective than implicit feature representations.

Inspired by previous works which show benefits of explicitly representing knowledge, we propose a text-entity interactive graph to model the relation between mentions and KB entities. As shown in Figure 2, the vertex set of this graph is made up of word nodes (*e.g.* yellow nodes) and entity nodes (*e.g.* green nodes). Word nodes represent the text information. We build inter-word edges connecting every sequential word in sentence. Once a mention (*e.g. pancreatic cancer*) is matched to an entity (*e.g. Pancreatic_Cancer*), we build two extra edges from corresponding entity node to the start and end word nodes of mention.

We apply GATs to model over the text-entity graph. GATs is a neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to enable (implicitly) specifying different weights to different nodes in a neighborhood. In our work, GATs can facilitate assigning different weights to different knowledge. For a multi-layers GATs, node representations are updated iteratively, each layer takes the output of previous layer and the final output of GATs is the output of the last layer. We start by initializing entity nodes with entity embeddings and word nodes with their hidden states in BiLSTM over the input sentence. Then, each layer is updated in the following process: Suppose the input to the i-th GATs layer is a set of node features $\{f_1, f_2, ..., f_{n+m} | f_i \in \mathbb{R}^F\}$ where F is the dimension of node features, n is the number of word nodes and m is the number of entity nodes. We have the adjacency matrix $A \in \mathbb{R}^{(n+m)\times(n+m)}$. The i-th GATs layer outputs a new set of node features $\{f'_1, f'_2, ..., f'_{n+m}\}$ by:

$$f'_i = \|_{k=1}^{K} \sigma(\sum_{j \in N_i} \alpha_{ij}^k W^k f_j) \quad (9)$$

$$\alpha_{ij}^k = \frac{exp(LeakyReLU(a[W^k f_i \| W^k f_j]))}{\sum_{k \in N_i} exp(LeakyReLU(a[W^k f_i \| W^k f_k]))} \quad (10)$$

where K is the number of attention heads, $\|$ represents concatenation, α_{ij}^k are normalized attention coefficients by k-th attention head, a is a weight vector and W^k is the correponding input linear transformation's weight matrix.

Specially, to update the final layer, we employ averaging, and delay applying the final nonlinearity:

$$f'_i = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^k W^k f_j) \quad (11)$$

where σ is the sigmoid function.

Inference

We concatenate the output of GATs and original word embeddings to form the input features of inference module $R = \{r_1, r_2, ..., r_n\}$. We adopt CRF to capture label dependencies by adding transition scores between neighboring labels. The probability of the ground-truth tag sequence Y is:

$$p(y|s) = \frac{exp(\sum_i (W^{y_i} r_i + T_{(y_{i-1}, y_i)}))}{\sum_{y'} exp(\sum_i (W^{y'_i} r_i + T_{(y'_{i-1}, y'_i)}))} \quad (12)$$

where T is the trainsition matrix and W^{y_i},$W^{y'_i}$ are learned parameters.Finally, the loss function of NER is defined as:

$$L = -\sum_{i=1}^{N} log(p(y_i|s_i)) \quad (13)$$

Experiments

Datasets The main experiments are conducted on five datasets from different domains, including spoken queries (MIT-Movie, MIT-Restaurant), defense and security (RE3D), anatomical (AnEM), and biomedical (BC5CDR-Disease).

- MIT-Movie (Liu et al. 2013b) contains 10343 movie queries where long constituents are annotated such as a movie's origin and plot descriptions. The size of type set is 12 including `Actor`, `Character`, `Director`, *etc.*
- MIT-Restaurant (Liu et al. 2013a) contains 7,136 restaurant queries. The dataset is annotated with 8 types including `Restaurant_Name`, `Amenity`, `Cuisine`, *etc.*
- RE3D (DSTL 2017) consists of 1,009 sentences and is relevant to the defence and security analysis domain. It contains 10 kinds of entities, including `Document_Reference`, `Location`, `Military_Platform`, *etc.*

Methods	Metrics	MIT-movie	RE3d	MIT-rest	BC5CDR	AnEM	Average
BiLSTM-CRF	P	76.78±0.35	67.68±0.44	79.96±0.28	83.42±1.30	75.41±1.13	76.65±0.70
	R	71.82±0.78	67.46±0.43	79.25±0.32	84.62±0.97	51.75±0.54	70.98±0.61
	F1	74.22±0.30	67.66±0.16	79.60•±0.23	84.02±0.35	61.38±0.31	73.38±0.30
BERT-CRF	P	76.57±0.42	63.56±0.36	79.36±0.30	84.00±1.09	64.46±1.11	73.59±0.66
	R	70.30±0.11	75.00 ±0.49	78.30±0.48	86.35±1.00	61.47±0.46	74.28±0.51
	F1	73.30±0.23	68.81•±0.18	78.82±0.35	85.16•±0.27	62.93•±0.29	73.80•±0.26
SoftGaz	P	77.07±0.36	66.06±1.30	78.46±0.32	86.33±0.70	66.15±1.13	74.81±0.76
	R	72.32±0.14	69.38±1.76	78.71±0.31	83.69±0.69	58.04±0.55	72.43±0.69
	F1	74.62•±0.20	67.68±0.26	78.59±0.31	84.99±0.35	61.83±0.32	73.54±0.29
KaNa (ours)	P	79.33±0.32	67.13±0.34	80.88±0.40	86.37±0.68	70.94±0.24	76.93±0.40
	R	73.64±0.28	71.60±0.33	79.98±0.32	85.89±0.66	57.32±0.29	73.69±0.38
	F1	**76.38±0.06**	**69.29±0.12**	**80.43±0.19**	**86.13±0.12**	**63.41±0.09**	**75.13±0.12**

Table 1: Experiment results of all methods on the five dataset. The best performance is highlighted in boldface. • indicates the best performing baselines. ± means the value plus/minus Standard Deviation.

- AnEM (Ohta et al. 2012) is a dataset manually annotated from 500 documents for anatomical entity mentions using a fine-grained classification system. The corpus annotation covers mentions of both healthy and pathological anatomical entities, such as `Kidney`, `Muscle` and `Blood`.
- BC5CDR-Disease (Li et al. 2016) is a collection of 1,500 PubMed titles and abstracts. It was used in the BioCreative V chemical disease relation. It contains one type of NEs: `Disease`.

Baselines We compare our model with three representative and state-of-the-art models.

BiLSTM-CRF: BiLSTM-CRF is a state-of-the-art NER model without considering knowledge incorporation. The BiLSTM effectively captures the sequential relationships amongst the input tokens and the CRF permits optimal, joint prediction of all the labels in the sentence, capturing the relationships at label level. We implement BiLSTM-CRF by using NCRF++(Yang and Zhang 2018).

BERT-CRF: BERT captures implicit language information from unlabeled text, achieving state-of-the-art results on many NLP tasks including NER. We use the BERT (Devlin et al. 2019) model pretrained on English Wikipedia and BookCorpus as a baseline. On top layer, a CRF module is employed as the inference layer.

SoftGaz: SoftGaz (Rijhwani et al. 2020) is a state-of-the-art knowledge-aware NER model. SoftGaz inserts the KB type features via an auto-encoder under a task specific KB setting. In our implementation, instead of manually resolving type heterogeneity, we use a hard filter method where only entities having type names appearing in intersection of two type systems are kept.

Implementation For hyper-parameter settings, the hidden state size of BiLSTM is 200. The dropout rate is set as 0.2 and 0.5 for BiLSTM output and pretrained embedding respectively. We apply a two-layer GATs for knowledge injection. The first layer consists of $K = 5$ attention heads. We set $F = 30$ for the node dimension of first layer. The last layer is a single attention head where $F = C$ (where C is the number of types in NER). The dropout rate is set to 0.1 and 0.4 for GATs layers and pretrained entity embedding respectively. We use the SGD optimizer in a mini-batch size of 10 with learning rate $\gamma = 1 \times 10^{-3}$ and L_2 regularization $\lambda = 0.005$. We use Wikidata (Vrandečić and Krötzsch 2014) as our KB. Wikidata is a free and open sourced knowledge base, which covers tens of thousands of entities with types over many domains. For the nodes initialization of text-entity graph, we use Entity-GloVe embeddings (Mikolov 2013). We also explore using PBG (Lerer et al. 2019) (*i.e.* pre-trained graph embeddings) as nodes initialization in ablation study.

In experiment, we leverage KnowBERT (Peters et al. 2019), a powerful knowledge-aware language model to generate every mention-entity pair.The evaluation metric for all experiments on various datasets is entity-level F1.

Result and Analysis

Overall Performance Table 1 presents the performance of our proposed model KaNa and compared baselines. From the table, we have several observations:

- In all cases, our proposed model KaNa outperforms all baselines. In particular, compared to the strongest baseline (BERT-CRF), KaNa improves F1 1.33 on average. Comparisons with knowledge free models (*i.e.* BiLSTM-CRF, BERT-CRF) validate the effectiveness of using KB knowledge in NER and comparisons with knowledge aware model (*i.e.* SoftGaz) show that our model has a better way of incorporating knowledge.
- Compared to those knowledge free models, KaNa leads to an $F1$ 1.75 improvement on average over BiLSTM-CRF. The recall is increased by 2.71% on average, this improvement could be brought by the extra entity information of KB. On the other hand, since BERT learns from unstructured text, it could lack structured knowledge. Our model learn from more accurate knowledge and outperform BERT-CRF by 3.34% on average precision.
- SoftGaz is a strong baseline under a task specific KB setting. However, without manually resolving type system heterogeneity, SoftGaz perform worse than BiLSTM-CRF on MIT-rest where a precision decline of 1.5% is

Method	P	R	F1
KaNa w/o Denoising	76.71	72.68	74.64
KaNa w Denoising	79.33	73.64	**76.38**

Table 2: Performance comparison of knowledge denoising with and without denoising module(Results reported on MIT-movie).

Method	P	R	F1
KaNa (Entity-Glove)	79.33	73.64	**76.38**
KaNa (PBG)	78.29	73.23	75.68
KaNa (One-Hop)	77.69	73.54	75.56

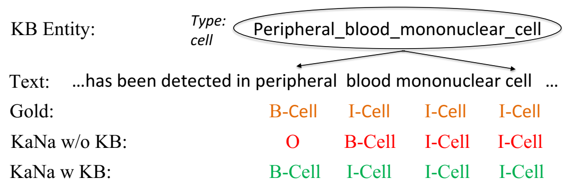
Table 3: Different ways of using selected knowledge (Results reported on MIT-movie). *Entity-Glove*: use Glove to initialize entity nodes. *PBG*: use translation embeddings (*e.g.* TransE) to initialize entity nodes. *One-Hop*: introducing one-hop knowledge (the entity with its corresponding relation and tail entity).

observed. It could probably due to type-compound noise since many restaurant names are ambiguous. KaNa outperforms SoftGaz by 1.59%, showing the effectiveness of knowledge denoising and injection module.

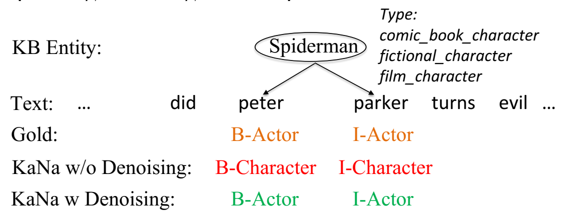
The Effect of of Knowledge We investigate the effectiveness of knowledge denoising by comparing KaNa models with and without denoising module. From the Table2, we can observe that KaNa model with knowledge denoising performs better than that without knowledge denoising. Those gains could be brought by filtering out task-irrelevant, incorrect type-compound entities.

Moreover, we explore what and how to use selected knowledge. The different ways of using selected KB knowledge in text-entity graph can fall into two aspects including what to use and how to use. Firstly, except for entity names and types, relational knowledge is also contained in KB. One selected entity can have certain relations with another in KB, which we call one-hop knowledge. So we implement KaNa (One-Hop) model to investigate whether NER benefits from such relational knowledge. Results in Table 3 show a decline of 0.82% on F1. The reason could be that one-hop knowledge brought by selected entities is not filtered, thus more noise than useful information is introduced. Secondly, for the entity nodes initialization, instead of infusing nodes with semantic information (Entity-Glove), we could also use KB representations (*i.e.* PBG) to provide more structural information. The use of PBG slightly decreases KaNa by 0.70%. It could be deduced that after using structural KB knowledge in the candidate generation and denosing phases, the semantic information of entities themselves dominates in the following use of KB knowledge.

Case Study We conduct case study to investigate the influence of using KB and how the knowledge denoising module impacts upon NER. The first case compares KaNa with and without knowledge base, displayed in Figure 3a. In case 1,


KB Entity:
Type: cell
Peripheral_blood_mononuclear_cell
Text: ...has been detected in peripheral blood mononuclear cell ...
Gold: B-Cell I-Cell I-Cell I-Cell
KaNa w/o KB: O B-Cell I-Cell I-Cell
KaNa w KB: B-Cell I-Cell I-Cell I-Cell

(a) Case 1: KaNa w/o KB means using BiLSTM-CRF to predict labels without considering knowledge base, KaNa w KB means incorporating knowledge base to predict labels.


Type: comic_book_character fictional_character film_character
KB Entity:
Spiderman
Text: ... did peter parker turns evil ...
Gold: B-Actor I-Actor
KaNa w/o Denoising: B-Character I-Character
KaNa w Denoising: B-Actor I-Actor

(b) Case 2: KaNa w/o Denoising means predicting labels by directly incorporating raw knowledge base without knowledge denoising. KaNa w Denoising means predicting labels by utilizing a refined knowledge base after knowledge denoising.

Figure 3: Case Study. Case 1 is selected from the test file of AnEM dataset. Case 2 belongs to the test file of MIT-Movie.

there is an entity *Peripheral_blood_mononuclea_cell* linked to the mention *Peripheral blood mononuclea cell*. Without the linked entity, the model can not integrate the correct boundary information and predict the label of *Peripheral* as *O*. With the linked entity, our model can correctly recognize the mention boundary. It illustrates the need of introducing KB knowledge. In case 2, we compare KaNa with and without denoising module. The mention *peter parker* is an actor, but is linked to his character *Spiderman*. Without denoising module, incorrect KB information misleads model to predict wrong type. By adding denoising module, such noise is filtered out and our model predicts the correct type `Actor`.

Conclusion and Future Work

In this paper, we proposed a novel knowledge-aware NER framework (KaNa), which is the first model using type-heterogeneous knowledge to improve NER performance. Type heterogeneity issues are alleviated via an unsupervised denoising method. Experiments on five challenging datasets demonstrate the effectiveness and generalization of our proposed model. As we described in ablation study, a straightforward way of using relational knowledge in NER couldn't lead to a performance improvement. Thus, in the future, we would like to investigate a more sophisticated KB knowledge incorporating mechanism, which can take more entity information, such as entity descriptions and relations, into consideration. We believe that, with a strong enough knowledge incorporating method, a unified multi-domain NER framework could be achieved by leveraging multi-domain KB knowledge.

References

Chen, H.; Lin, Z.; Ding, G.; Lou, J.; Zhang, Y.; and Karlsson, B. 2019. GRN: Gated Relation Network to Enhance

Convolutional Neural Network for Named Entity Recognition. In *Proceedings of AAAI*, 6236–6243.

Cui, L.; and Zhang, Y. 2019. Hierarchically-Refined Label Attention Network for Sequence Labeling. In *Proceedings of IJCNLP*, 4113–4126.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*, 4171–4186.

Ding, R.; Xie, P.; Zhang, X.; Lu, W.; Li, L.; and Si, L. 2019. A Neural Multi-digraph Model for Chinese NER with Gazetteers. In *Proceedings of ACL*, 1462–1467.

DSTL. 2017. Relationship and Entity Extraction Evaluation Dataset. https://github.com/dstl/re3d. Accessed January, 2020.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of SIGKDD*, 855–864.

Gui, T.; Zou, Y.; Zhang, Q.; Peng, M.; Fu, J.; Wei, Z.; and Huang, X. 2019. A Lexicon-Based Graph Neural Network for Chinese NER. In *Proceedings of IJCNLP*, 1040–1050.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of NIPs*, 1024–1034.

He, Q.; Wu, L.; Yin, Y.; and Cai, H. 2020. Knowledge-Graph Augmented Word Representations For Named Entity Recognition. In *Proceedings of AAAI*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv: Computation and Language* .

Kazama, J.; and Torisawa, K. 2007. Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *Proceedings of ACL*, 698–707.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition 260–270. ISSN 1045-9227. doi:10.18653/v1/N16-1030. URL http://arxiv.org/abs/1603.01360.

Lerer, A.; Wu, L.; Shen, J.; Lacroix, T.; Wehrstedt, L.; Bose, A.; and Peysakhovich, A. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of SysML*.

Li, J.; Sun, Y.; Johnson, R. J.; Sciaky, D.; Wei, C.; Leaman, R.; Davis, A. P.; Mattingly, C. J.; Wiegers, T. C.; and Lu, Z. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database J. Biol. Databases Curation* 2016. doi:10.1093/database/baw068. URL https://doi.org/10.1093/database/baw068.

Li, Y.; Bontcheva, K.; and Cunningham, H. 2005. Named Entity Recognition using an HMM-based Chunk Tagger. In *Proceedings of NAACL*, 319–339.

Li, Y.; Bontcheva, K.; and Cunningham, H. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of CoNLL*, 147–155.

Liu, J.; Pasupat, P.; Cyphers, S.; and Glass, J. 2013a. Asgard: A portable architecture for multilingual dialogue systems. In *Proceedings of ICASSP*, 8386–8390.

Liu, J.; Pasupat, P.; Cyphers, S.; and Glass, J. 2013b. Query understanding enhanced by hierarchical parsing structures. In *Proceedings of ICASSP*, 72–77.

Lu, P.; Bai, T.; and Langlais, P. 2019. SC-LSTM: Learning Task-Specific Representations in Multi-Task Learning for Sequence Labeling. In *Proceedings of NAACL*, 2396–2406.

Luo, G.; Huang, X.; Lin, C.; and Nie, Z. 2015. Joint Entity Recognition and Disambiguation. In *Proceedings of EMNLP*, 879–888.

Ma, X.; and Hovy, E. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of ACL*, 1064–1074.

Mikolov, T. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPs*, 3111–3119.

Morgan, A. A.; Hirschman, L.; Yeh, A. S.; and Colosimo, M. E. 2003. Gene Name Extraction Using FlyBase Resources. In *Proceedings of ACL*, 1–8.

Ngo, Q. H.; Dien, D.; and Winiwarter, W. 2014. Building English-Vietnamese named entity corpus with aligned bilingual news articles. In *Proceedings of WSSANLP*, 85–93.

Nothman, J.; Ringland, N.; Radford, W.; Murphy, T.; and Curran, J. R. 2013. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence* 194: 151–175. doi:10.1016/j.artint.2012.03.006. URL https://doi.org/10.1016/j.artint.2012.03.006.

Ohta, T.; Pyysalo, S.; Tsujii, J.; and Ananiadou, S. 2012. Open-domain Anatomical Entity Mention Detection. In *Proceedings of ACL*, 27–36.

Passos, A.; Kumar, V.; and Mccallum, A. 2014. Lexicon Infused Phrase Embeddings for Named Entity Resolution. In *Proceedings of CONLL*, 78–86.

Perozzi, B.; Alrfou, R.; and Skiena, S. 2014. DeepWalk: online learning of social representations. In *Proceedings of SIGKDD*, 701–710.

Peters, M. E.; Ammar, W.; Bhagavatula, C.; and Power, R. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL*, 1756–1765.

Peters, M. E.; Neumann, M.; IV, R. L. L.; Schwartz, R.; Joshi, V.; Singh, S.; and Smith, N. A. 2019. Knowledge Enhanced Contextual Word Representations. In *Proceedings of EMNLP-IJCNLP*, 43–54.

Radford, W.; Carreras, X.; and Henderson, J. 2015. Named entity recognition with document-specific KB tag gazetteers. In *Proceedings of EMNLP*, 512–517.

Richman, A. E.; and Schone, P. 2008. Mining Wiki Resources for Multilingual Named Entity Recognition. In *Proceedings of ACL*, 1–9.

Rijhwani, S.; Zhou, S.; Neubig, G.; and Carbonell, J. 2020. Soft Gazetteers for Low-Resource Named Entity Recognition. In *Proceedings of ACL*.

Strubell, E.; Verga, P.; Belanger, D.; and Mccallum, A. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of EMNLP*, 2670–2680.

Sui, D.; Chen, Y.; Liu, K.; Zhao, J.; and Liu, S. 2019. Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network. In *Proceedings of IJCNLP*, 3828–3838.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of WWW*, 1067–1077.

Tran, Q.; Mackinlay, A.; and Yepes, A. J. 2017. Named Entity Recognition with Stack Residual LSTM and Trainable Bias Decoding. In *Proceedings of IJCNLP*, 566–575.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of ICLR*, 4171–4186.

Vlachos, A.; and Gasperin, C. 2006. Bootstrapping and Evaluating Named Entity Recognition in the Biomedical Domain. In *Proceedings of NAACL*, 138–145.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10): 78–85.

Yang, J.; and Zhang, Y. 2018. NCRF++: An Open-source Neural Sequence Labeling Toolkit. In *Proceedings of ACL*. URL http://aclweb.org/anthology/P18-4013.

Zhou, Z.; and Zhang, M. 2006. Multi-Instance Multi-Label Learning with Application to Scene Classification. In *Proceedings of NIPs*, 1609–1616.