Relation Extraction with Type-aware Map Memories of Word Dependencies

Guimin Chen♦*, Yuanhe Tian♥*, Yan Song♠♥†, Xiang Wan♦
♦QTrade  ♥University of Washington  ♠The Chinese University of Hong Kong (Shenzhen)
†Shenzhen Research Institute of Big Data
chenguimin@foxmail.com  yhtian@uw.edu  songyan@cuhk.edu.cn  wanxiang@sribd.cn

Abstract

Relation extraction is an important task in information extraction and retrieval that aims to extract relations among the given entities from running texts. To achieve a good performance for this task, previous studies have shown that a good modeling of the contextual information is required, where the dependency tree of the input sentence can be a beneficial source among different types of contextual information. However, most of these studies focus on the dependency connections between words with limited attention paid to exploiting dependency types. In addition, they often treat different dependency connections equally in modeling so that suffer from the noise (inaccurate dependency parses) in the auto-generated dependency tree. In this paper, we propose a neural approach for relation extraction, with type-aware map memories (TaMM) for encoding dependency types obtained from an off-the-shelf dependency parser for the input sentence. Specifically, for each word in an entity, TaMM maps all associated words along with the dependencies among them to memory slots and then assigns a weight to each slot according to its contribution to relation extraction. Our approach not only leverages dependency connections and types between words, but also distinguishes reliable dependency information from noisy ones and appropriately model them. The effectiveness of our approach is demonstrated by the experiments on two English benchmark datasets, where our approach achieves state-of-the-art performance on both datasets.1

1 Introduction

Relation extraction is an important natural language processing (NLP) task that facilitates information extraction, whose results is beneficial to downstream tasks such as schema induction (Nimishakavi et al., 2016), knowledge graph construction (Yu et al., 2017), and question answering (Xu et al., 2016). Normally, relation extraction aims to predict the relation between each pair of entities in a given sentence. For example, in the sentence “the [bone marrow]e1 produces [stem cells]e2” with the entity terms “bone marrow” and “stem cells”, the relation between the two entities is “Product-Producer”. Therefore, the ability of modeling the context from the input is of great importance to guarantee the performance of relation extraction. To this end, approaches based on neural networks have achieved promising success for the task in the past decade (Socher et al., 2012; Zeng et al., 2014; Zhang and Wang, 2015; Xu et al., 2015; dos Santos et al., 2015; Zhang et al., 2015; Wang et al., 2016; Zhou et al., 2016; Zhang et al., 2017; Wu and He, 2019; Soares et al., 2019; Fu et al., 2019; Aydar et al., 2020; Tian et al., 2021c) because of their effectiveness in capturing contextual information by powerful encoders.

In addition, previous studies try to improve relation extraction performance by incorporating extra knowledge into their models. Among all such knowledge, syntactic information from the auto-generated dependency parse of the input sentence indicates its helpfulness to improve model performance for the reason that word dependencies provide long distance contextual information (Xu et al., 2015). However, in previous studies, the main focus is the dependencies among words, with little attention paid to dependency types, which are also

---

1Equal contribution.
2Corresponding author.
3The code and models involved in this paper are released at https://github.com/cuhksz-nlp/RE-TaMM.
The evaluation of different models is performed on two English benchmark datasets, i.e., ACE2005 and SemEval 2010 Task 8 (Hendrickx et al., 2010), where our approach outperforms all baselines and “text classification task, where an input sentence produces a correct prediction for relations. Therefore, it is straightforward to consider integrating extra features to enhance contextual modeling. Of all such features, the syntactic information suggested by the dependency tree of the input sentence has been demonstrated to be useful for relation extraction in many studies (Xu et al., 2015; Zhang et al., 2018; Guo et al., 2019). However, most models to leverage the dependency information are not naturally appropriate to model the dependency types among words. It is required to find an appropriate approach to leverage the dependency type information.

Of all choices, key-value memory networks (KVMN) (Miller et al., 2016) is an effective solution in modeling pair-wisely organized information to improve many NLP tasks (Tapaswi et al., 2016; Das et al., 2017; Mino et al., 2017; Xu et al., 2019; Nie et al., 2020; Song et al., 2020; Tian et al., 2020a,d, 2021b). Specifically, KVMN maps the information instances into a list of memory slots $s_i = (k_i, v_i)$ ($i$ is the index of the memory slot $s_i$) with $k_i$ referring to the key and $v_i$ the value, respectively. The KVMN addresses the memory slot $s_i$ by assigning a weight $p_i$ to the value $v_i$ by comparing the input (denoted by $x$) to the key $k_i$:

$$p_i = \text{softmax} (\mathbf{A} \Phi_X (x) \cdot \mathbf{A} \Phi_K (k_i))$$

(1)

where $\Phi_.$ are functions that map the input features into their embeddings and $\mathbf{A}$ is a matrix that maps the embeddings into another vector space. After addressing all memory slots, KVMN reads the values by computing the weighted sum of the value vectors (i.e., $\mathbf{A} \Phi_V (v_i)$) using the resulting probability weights (i.e., $p_i$), which is expressed by

$$\mathbf{a} = \sum_j p_j \cdot \mathbf{A} \Phi_V (v_i)$$

(2)

Then, $\mathbf{a}$ is incorporated into the input representation by an element-wise summation:

$$\mathbf{o} = \mathbf{A} \Phi_X (x) + \mathbf{a}$$

(3)

Thus, the resulting vector $\mathbf{o}$ contains the weighted information from all values in the memory slots and is finally used to predict the output.

---

2For example, in Figure 1, the dependency between “the” and “marrow” contributes less than the dependency between “marrow” and “produces” to relation extraction.

3$E_1$ and $E_2$ are actually sub-strings of $X$ and we assume $E_1$ is on the left side of $E_2$. 

---

2502
3 The Proposed Approach

Although KVMN can be used to leverage extra information for relation extraction, it loses the information of keys by using it as a weighting component as stated previously. Therefore, we propose type-aware map memories (TaMM) to leverage both context words (keys) and dependency types (values) to improve relation extraction, where two types of dependency information, i.e., “in-entity” and “cross-entity” dependencies are considered.

Figure 2 illustrates the architecture of our approach, in which the entities in the input $\mathcal{X}$ is highlighted in red; the left part illustrates the backbone classification model; the right part shows the process to leverage the in-entity and cross-entity memory slots associated with “bone” (highlighted in yellow) through the proposed type-aware map memories (TaMM). In entity and cross-entity memory slots are written in blue and green color respectively.

The following texts illustrates the details of our proposed approach, including how we construct the memory slots and the computation of TaMM, with its application in relation extraction.

3.1 Memory Slot Construction

In order to construct the memory slots used in our approach, we firstly use an off-the-shelf toolkit to generate the dependency parsing results of the input $\mathcal{X}$. In the parse tree, every word in $\mathcal{X}$ is connected with its governor and its dependents with labeled dependency connections; for any two words in $\mathcal{X}$, there is exactly one path between them\(^4\). For each word in an entity, e.g., the word $x_{iu}$ in $E_u$ ($i_u$ is the index of $x_{iu}$ in $\mathcal{X}$ and $u \in \{1, 2\}$), we consider two types of dependency information suggested by the obtained dependency tree of $\mathcal{X}$ and construct their corresponding memory slots. The first one is “in-entity” memory slots constructed upon all the governor and dependents of $x_{iu}$ (i.e., first-order dependencies). The second is “cross-entity” memory

\(^4\)The dependency parsing results actually build a graph (tree) of the input $\mathcal{X}$, where words in $\mathcal{X}$ represent the graph nodes, and the dependency connections are the graph edges.
we find its dependent "marrow" (i.e., "stem cells") where (i.e., "compound"

Therefore, we obtain a list of memory slots with the directional information of the dependency type.

Cross-entity Memory Slots

Cross-entity memory slots for it should be if the word we focus on is "stem cells" and obtain the dependency slot dependency relation type "compound"

For example, the head of "bone marrow" is "marrow" and the head of "stem cells" is "cells".

In summary, for $x_i$ in in $E_u$, we obtain the in-entity memory slot list $S_{in}^{(in)}$ and the cross-entity memory slot list $S_{cross}^{(cross)}$, which are fed into the TaMM module as illustrated in Figure 2.

3.2 Type-aware Map Memories

There are previous approaches for relation extraction that leverage dependency information and focus on dependencies among words without considering their dependency types. With learning from such information, there is a nonnegligible challenge that there are noises in the auto-generated depen-

---

5. The other entity means $E_2$ if $x_i$ is in $x_i$.

6. We add a " mark before the dependency type to illustrate the directional information of the dependency type.
To address the aforementioned limitations in KVMN, we propose type-aware map memories (TaMM) to incorporate the dependency information carried by both the keys and values (i.e., the memory slots), where the architecture of TaMM is illustrated on the top right of Figure 2. Specifically, for each word in an entity, e.g., the word \( x_{iu} \), in \( E_u \) (\( i_u \) is the index of \( x_{iu} \) in \( X \) and \( u \in \{ 1, 2 \} \)), we consider two types of dependency information, i.e., “in-entity” and “cross-entity” dependency information, and construct their corresponding memory slots. We denote the \( j \)-th in-entity and cross-entity memory slots as \( s_{iu,j}^{(in)} = (k_{iu,j}^{(in)}, v_{iu,j}^{(in)}) \) and \( s_{iu,j}^{(cross)} = (k_{iu,j}^{(cross)}, v_{iu,j}^{(cross)}) \), respectively, and use the same process to model them.

Taking the in-entity memory slots as an example, we firstly use two matrices to map the keys \( k_{iu,j}^{(in)} \) and values \( v_{iu,j}^{(in)} \) in the memory slots into their embeddings, which are denoted by \( e_{iu,j}^{k,(in)} \) and \( e_{iu,j}^{v,(in)} \), respectively. Next, we compute the weight \( p_{iu,j} \) assigned for each value through the inner production between the key embedding \( e_{iu,j}^{k,(in)} \) and the hidden vector of \( x_{iu} \) (which is denoted as \( h_{iu} \)) obtained from the encoder in the backbone model:

\[
p_{iu,j} = \frac{\exp \left( h_{iu} \cdot e_{iu,j}^{k,(in)} \right)}{\sum_{j=1}^{m_{iu}^{(in)}} \exp \left( h_{iu} \cdot e_{iu,j}^{k,(in)} \right)} \tag{5}
\]

where \( m_{iu}^{(in)} \) is the number of in-entity memory slots associated with \( x_{iu} \). Then, we apply the weights to the corresponding memory slots and obtain the weighted sum (denoted as \( a_{iu}^{(in)} \)) of both keys and values through

\[
a_{iu}^{(in)} = \sum_{j=1}^{m_{iu}^{(in)}} p_{iu,j} (e_{iu,j}^{k,(in)} + e_{iu,j}^{v,(in)}) \tag{6}
\]

where “+” refers to element-wise sum of vectors. Therefore, compared to KVMN, our approach is able to leverage both context words and dependency types associated with \( x_{iu} \).

With the same process for in-entity memory slots, we deal with the cross-entity ones and obtain the weighted sum \( a_{iu}^{(cross)} \). Finally, we concatenate the two resulting vectors by

\[
a_{iu} = a_{iu}^{(in)} \oplus a_{iu}^{(cross)}
\]

with \( a_{iu} \) denoting the output of TaMM and containing the weighted dependency information to enhance the backbone model.

### 3.3 Relation Extraction with TaMM

Once the TaMM is built, it is straightforward to apply it to relation extraction through a backbone classifier. In our approach, we use BERT (Devlin et al., 2019) as the classifier to encode the input \( X \) and obtain the hidden vectors for all words. Note that we only use the hidden vectors of the words in the two entities to predict their relations. Therefore, for each word \( x_{iu} \) in the entity \( E_u \), we feed \( h_{iu} \) into TaMM and obtain the corresponding output \( a_{iu} \). Then, we concatenate \( h_{iu} \) and \( a_{iu} \), and for each entity \( E_u \), use the max pooling strategy to obtain the vectorized representation \( o_u \) by

\[
o_u = \text{MaxPooling}(h_{iu} \oplus a_{iu}) \tag{7}
\]

Afterwards, we concatenate the representation of the two entities (i.e., \( o_1 \) for \( E_1 \) and \( o_2 \) for \( E_2 \)) and pass the resulting vector through a fully connected layer (a classifier) to obtain the final prediction \( \hat{y} \) by

\[
\hat{y} = W \cdot (o_1 \oplus o_2) + b \tag{8}
\]

where \( W \) and \( b \) are the trainable weight matrix and bias vector for the fully connected layer.

### 4 Experimental Settings

#### 4.1 Datasets

Two English benchmark datasets, i.e., ACE2005EN (ACE2005) and SemEval 2010 Task 8 (SemEval) (Hendrickx et al., 2010) are used in the experiments to evaluate our approach. For ACE2005, we follow the same preprocessing as that in Christopoulou

---

*https://catalog.ldc.upenn.edu/LDC2006T05.  
*http://docs.google.com/View?docid=dfvxd49s36c28v9pmw.

---

| # Instances | ACE2005 | SemEval |
|-------------|---------|---------|
| Train       | 48,198  | 8,000   |
| Dev         | 11,854  | -       |
| Test        | 10,097  | 2,717   |

Table 1: The statistics (number of instances and relation types) of the two benchmark datasets.
Table 2: The hyper-parameters tested in tuning our models. The best ones used in our final experiments are highlighted in boldface.

| Hyper-parameters | Values |
|------------------|--------|
| Learning Rate    | $5e - 6, 1e - 5, 2e - 5, 3e - 5$ |
| Warmup Rate      | 0.06, 0.1 |
| Dropout Rate     | 0.1 |
| Batch Size       | 16, 32, 64, 128 |

Table 3: F1 scores of our TaMM and baselines (i.e., BERT, standard GCN, standard GAT, and KVMN) on the test sets of ACE2005 and SemEval, where BERT-base and BERT-large encoders are used. For KVMN and TaMM, different combinations of in-entity and cross-entity dependency information (i.e., in-entity only, cross-entity only, and both of them) are tried.

| Models            | ACE2005 | SemEval |
|-------------------|---------|---------|
| BERT-base         | 75.31   | 87.87   |
| + GCN             | 75.59   | 88.19   |
| + GAT             | 76.01   | 88.39   |
| + KVMN (In)       | 76.40   | 88.73   |
| + TaMM (In)       | 76.80   | 89.11   |
| + KVMN (Cross)    | 76.45   | 88.61   |
| + TaMM (Cross)    | 76.61   | 88.74   |
| + KVMN (Both)     | 76.83   | 89.98   |
| + TaMM (Both)     | 77.07   | 89.18   |
| BERT-large        | 76.79   | 89.02   |
| + GCN             | 77.17   | 89.43   |
| + GAT             | 77.23   | 89.39   |
| + KVMN (In)       | 77.32   | 89.42   |
| + TaMM (In)       | 77.76   | 89.72   |
| + KVMN (Cross)    | 77.21   | 89.37   |
| + TaMM (Cross)    | 77.66   | 89.58   |
| + KVMN (Both)     | 77.96   | 89.88   |
| + TaMM (Both)     | 78.98   | 90.06   |

5 Results and Analyses

5.1 Overall Performance

In the main experiments, we run our models using BERT-base and BERT-large encoders with and without TaMM and try different combinations of in-entity and cross-entity dependency information (i.e., in-entity dependency information only, cross-entity dependency information only, and both of them). We also run the baselines using standard graph convolutional networks (GCN), standard graph attention networks (GAT), and KVMN to leverage the dependency information. Table 3 shows the results (F1 scores) of different models.\footnote{We use the official evaluation script downloaded from http://semeval2.fbk.eu/scorers/task08/SemEval2010_task8_scorer-v1.2.zip.}

We use the dataset split from https://github.com/tticoin/LSTM-ER/tree/master/data/ace2005/split.\footnote{We use the dataset split from https://github.com/huggingface/transformers/.}

We use SCT under version 3.9.2 from https://stanfordnlp.github.io/CoreNLP/.\footnote{We use SCT under version 3.9.2 from https://stanfordnlp.github.io/CoreNLP/.

We download different BERT models from https://github.com/huggingface/transformers.\footnote{We download different BERT models from https://github.com/huggingface/transformers.}

We use the evaluation script from sklearn framework.\footnote{We use the evaluation script from sklearn framework.}
There are several observations. First, TaMM works well with both BERT base and large, where consistent improvement is observed over the BERT-base and BERT-large baselines across all datasets, although they have already achieved very good performance. Second, TaMM outperforms standard GCN and GAT models, which can be attributed to our modeling of dependency type information in TaMM. Third, under all the three settings to incorporate different types of dependency information (i.e., in-entity, cross-entity, and both), our models with TaMM outperforms the BERT baseline and the highest F1 score is achieved when both in-entity and cross-entity dependency information are used (i.e., + TaMM (Both)). This observation confirms the individual contribution of in-entity and cross-entity dependency information as well as the effectiveness of our approach to leverage them together to improve model performance. Fourth, compared with our TaMM models using cross-entity dependency information only (i.e., + TaMM (Cross)), the models using in-entity dependency information only (i.e., + TaMM (In)) achieves higher results in most cases. One possible explanation could be the following. There are overlaps between in-entity dependencies and cross-entity dependencies. For example, the dependency between “bone” and “marrow” is shared by both in-entity dependencies and cross-entity dependencies in Figure 3. Therefore, with in-entity dependency only, TaMM not only leverages the contextual words directly associated with the entities themselves, but also can still partially benefit from the contextual information along the dependency path, whereas TaMM with cross-entity dependency only fails to leverage the contextual words directly associated with the entities, which leads TaMM (In) to achieve better performance than TaMM (Cross). Fifth, for all the settings, our model with TaMM consistently outperforms the baselines with KVMN, which demonstrates the effectiveness of our approach to improve relation extraction. The explanation is that TaMM is able to leverage both context words (keys) and dependency types (values) at the same time, while KVMN fails to incorporate the context information carried by keys, which leads KVMN to omit some important features and thus get inferior results. Moreover, we compare our model under the best setting (i.e., the ones using TaMM to leverage both in-entity and cross-entity dependency relation) with previous studies and report the results (F1 scores) in Table 4. It is found that our model with BERT-large encoder outperforms all previous studies (including the ones also using BERT-large encoder).

5.2 The Effect of Dependency Information

To analyze the effect of using dependency information, we perform three investigations on models using BERT-large encoder.

The first investigation is to examine different orders of dependencies used in TaMM. Previous experiments showed the effectiveness of our model with TaMM on first-order word dependencies. We also try second- and third-order dependencies via the model (i.e., large BERT) with TaMM (Both). The results (with scores from the first-order dependencies) are reported on table 5, where the corresponding results from the models with TaMM (In) as well as the BERT-large baseline are also reported. The observations are drawn as follows. First, models with TaMM under all settings outperform the BERT-large baseline, which is confirmed by all results on both datasets. Second, models with TaMM (Both) consistently outperform the ones with TaMM (In) under the same setting, which indicates the cross-entity dependencies are able to bring greater improvements. Third, for models

Table 4: The comparison between our models (the ones using TaMM (Both)) and previous studies on ACE2005 and SemEval. Models with dependency features and BERT-large are marked by “†” and “*”, respectively.

| Models                  | ACE2005 | SemEval |
|-------------------------|---------|---------|
| Wang et al. (2016)      | -       | 88.0    |
| Zhou et al. (2016)      | -       | 84.0    |
| Zhang et al. (2018)     | 64.2    | -       |
| Ye et al. (2019)        | 68.9    | -       |
| *Wu and He (2019) (BERT-large) | - | 89.2 |
| *Soares et al. (2019) (BERT-large) | - | 89.5 |
| Sun et al. (2020)       | -       | 86.0    |
| Yu et al. (2020)        | -       | 86.4    |
| Mandy et al. (2020)     | 85.9    |         |

*TaMM (Both) (BERT-base) 77.07 89.18
*TaMM (Both) (BERT-large) 78.98 90.06

Table 5: F1 scores of models using BERT-large and TaMM (In/Both) to leverage 1st-, 2nd-, and 3rd-order dependencies. “N/A” refers to no order can be applied.
with TaMM (Both), using higher order dependencies often results in inferior results; while the trend is on the opposite for models with TaMM (in). One possible explanation is that for TaMM (both), most essential word dependencies in between the two entities have already been encoded, higher order dependencies sometimes introduce noise other than useful information; while for TaMM (In), leveraging higher order dependencies allows the model to cover more contextual information along the dependency path between two entities.

The second is to explore the performance of our model on different test instances grouped by their entity distance (i.e., the number of words between the two entities), to see whether our approach can capture long-distance word-word dependencies and help with relation extraction. In doing so, we split the test set of SemEval into three groups according to the entity distance (i.e., from 0 to 4, from 5 to 9, and higher than 10) and perform our best TaMM model and the BERT baseline on them. Figure 4 illustrates the performance of TaMM (i.e., the orange bar) and BERT (i.e., the blue bar). It can be found that our TaMM outperforms the BERT-baseline on all three groups of test instances, where bigger gaps can be observed when the entities’ distance goes higher. This observation demonstrates the effectiveness of our approach to encode dependency information to improve relation extraction.

The third investigation is to explore the effect of TaMM using different dependency parsers. Specifically, in addition to the Stanford CoreNLP Toolkits (SCT) used in the main experiments, we also try spaCy\(^\text{16}\) to obtain the dependency trees and report the results (with BERT-large encoder) in Table 6. It is found that models with different dependency parsers consistently outperform the BERT-large baseline, which indicates the robustness of our model design in improving relation extraction.

\(^{16}\)https://spacy.io/

### 5.3 Case Study

To examine how TaMM leverages dependency information to improve model performance, in Figure 5, we show an example input where our approach successfully predicts the relation in between the two entities (in red colors) to be “Entity-Destination”, while the BERT-large baseline fails to do so (“Component-Whole”). In the figure, the dependencies between words are highlighted in different colors to represent the total weights assigned to their corresponding in-entity and cross-entity memory slots, where darker color refers to higher weight. Overall, we find that the most emphasized dependencies are along the dependency path connecting the two entities, where the memory slots for those dependencies receive the highest weights. For the first entity “treadmill”, the dependency type `nsubj: pass` (passive nominal subject) in the highlighted memory slot (installed, `nsubj: pass`) suggests the first entity is the patient of the action install; similarly, for the second entity “space station”, the highlighted dependency type `obj` (object) suggests this entity is the location of the action install given the fact that the input is a passive sentence. Therefore, our approach is able to leverage these cues learned from word dependencies and their dependency types so as to predict the correct relation for the two entities: “Entity-Destination”.

### 6 Related Work

Relation extraction is an important task in NLP, which significantly relies on a good modeling of the contextual information to achieve outstanding model performance. To improve the capability of context modeling for relation extraction, studies in the past decade leverage neural networks, such as using CNN (Zeng et al., 2014; Wang et al., 2016), RNN (Socher et al., 2012; Xu et al., 2015; Zhou et al., 2016) and BERT encoders (Wu and He, 2019; Soares et al., 2019; Wang et al., 2019). To further
enhance the models for this task, incorporating extra knowledge into the models has been proved as an effective method, where normally three types of extra knowledge are used: lexical, syntactic and semantic knowledge, and syntactic knowledge has been proved to be useful for this task (Xu et al., 2015). With this finding, there are studies also using advanced neural architecture, such as graph convolutional networks, to incorporate syntactic knowledge from auto-generated dependency parse of the input sentence (Zhang et al., 2018; Guo et al., 2019; Sun et al., 2020; Yu et al., 2020; Mandya et al., 2020). Compared to the aforementioned studies, TaMM offers a simple yet effective non-graph-based approach to leverage dependencies for relation extraction. TaMM provides the ability not only incorporate both word dependencies and their types into the model to help improve relation extraction performance, but also discriminatively leverage the dependencies by assigning different weights to them, which can address the potential noise in the auto-generated dependencies and thus further improve model performance.

7 Conclusion

In this paper, we proposed an effective method for relation extraction with word dependencies encoded by TaMM, whose keys and values are built upon the dependency tree of the input sentence obtained from off-the-shelf toolkits. Particularly, for each entity in the sentence, we extract words associated with it according to the dependency parse of the input sentence and their corresponding dependency relation types. Then, we use TaMM to encode and weight such information and integrate it into the relation extraction task. The novelty of this work lies in the modeling of contextual information through dependencies and their relation types encoded in TaMM. Experimental results on two public English benchmark datasets illustrate the effectiveness of our approach with state-of-the-art performance achieved on all datasets.

Acknowledgements

This work is supported by Chinese Key-Area Research and Development Program of Guangdong Province (2020B0101350001) and NSFC under the project “The Essential Algorithms and Technologies for Standardized Analytics of Clinical Texts” (12026610). This work is also partially supported by Shenzhen Institute of Artificial Intelligence and Robotics for Society under the project “Automatic Knowledge Enhanced Natural Language Understanding and Its Applications” (AC01202101001).

References

Mehmet Aydar, Ozge Bozal, and Furkan Ozbay. 2020. Neural Relation Extraction: a survey. arXiv e-prints, pages arXiv–2007.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A Walk-based Model on Entity Graphs for Relation Extraction. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 81–88.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 358–365.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of
Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1409–1418.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 241–251.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarnuid O Séaghdha, Sebastian Pado, Marco Pennacchiotti, Lorezena Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency Based Embeddings for Sentence Classification Tasks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1490–1500, San Diego, California.

Yang Liu and Mirella Lapata. 2018. Learning Structured Text Representations. Transactions of the Association for Computational Linguistics, 6:63–75.

Angrosh Mandya, Danushka Bollegala, and Frans Cohen. 2020. Graph Convolution over Multiple Dependency Sub-graphs for Relation Extraction. In Proceedings of the 28th International Conference on Computational Linguistics, pages 6424–6435.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1400–1409.

Hideya Mino, Masao Utiyama, Eiichiro Sumita, and Takenobu Tokunaga. 2017. Key-value Attention Mechanism for Neural Machine Translation. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 290–295, Taipei, Taiwan.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving Named Entity Recognition with Attentive Ensemble of Syntactic Information. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4231–4245.

Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation Schema Induction using Tensor Factorization with Side Information. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.

Cícero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2895–2905.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning Word Representations with Regularization from Prior Knowledge. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 143–152.

Yan Song and Shuming Shi. 2018. Complementary Learning of Word Embeddings. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 4368–4374.

Yan Song, Shuming Shi, and Jing Li. 2018. Joint Learning Embeddings for Chinese Words and Their Components via Ladder Structured Networks. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 4375–4381.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing Medical Conversations via Identifying Important Utterances. In Proceedings of the 28th International Joint Conference on Computational Linguistics, pages 717–729.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaptation for N-gram Enhanced Text Encoders. arXiv preprint arXiv:2105.01279.

Kai Sun, Richong Zhang, Yongyi Mao, Samuel Mensah, and Xudong Liu. 2020. Relation Extraction with Convolutional Network over Learnable Syntax-Transport Graph. In AAAI, pages 8928–8935.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding Stories in Movies
Yuanhe Tian, Guimin Chen, and Yan Song. 2021a. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2910–2922, Online.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021b. Enhancing Aspect-level Sentiment Analysis with Word Dependencies. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 3726–3739, Online.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021c. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks. In Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020a. Improving Biomedical Named Entity Recognition with Syntactic Information. BMC Bioinformatics, 21:1471–2105.

Yuanhe Tian, Yan Song, and Fei Xia. 2020b. Joint Chinese Word Segmentation and Part-of-speech Tagging via Multi-channel Attention of Character N-grams. In Proceedings of the 28th International Conference on Computational Linguistics, pages 2073–2084.

Yuanhe Tian, Yan Song, and Fei Xia. 2020c. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6037–6044.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. 2020d. Improving Chinese Word Segmentation with Wordhood Memory Networks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8274–8285, Online.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting Multiple-Relations in One-Pass with Pre-Trained Transformers. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1371–1377.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation Classification via Multi-Level Attention CNNs. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1298–1307.

Shanchan Wu and Yifan He. 2019. Enriching Pre-trained Language Model with Entity Information for Relation Classification. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pages 2361–2364.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2937–2947, Minneapolis, Minnesota.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying Relations via Long Short Term Memory Networks Along Shortest Dependency Paths. In Proceedings of the 2015 conference on empirical methods in natural language processing, pages 1785–1794.

Wei Ye, Bo Li, Rui Xie, Zhonghao Sheng, Long Chen, and Shikun Zhang. 2019. Exploiting Entity BIO Tag Embeddings and Multi-task Learning for Relation Extraction with Imbalanced Data. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1351–1360.

Bowen Yu, Mengge Xue, Zhenyu Zhang, Tingwen Liu, Wang Yubin, and Bin Wang. 2020. Learning to Prune Dependency Trees with Rethinking for Neural Relation Extraction. In Proceedings of the 28th International Conference on Computational Linguistics, pages 3842–3852, Barcelona, Spain (Online).

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved Neural Relation Detection for Knowledge Base Question Answering. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 571–581.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation Classification via Convolutional Deep Neural Network. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation Classification via Recurrent Neural Network. arXiv preprint arXiv:1508.01006.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional Long Short-Term Memory Networks for Relation Classification. In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, pages 73–78.
Appendix A. Experimental Results on the Development Set

Table 7 reports the F1 scores of different models on the development set of ACE2005.\footnote{SemEval does not have an official dev set.}

| Models    | BERT-base | BERT-Large |
|-----------|-----------|------------|
| BERT      | 75.03     | 76.51      |
| + GCN     | 75.33     | 76.82      |
| + GAT     | 75.77     | 76.89      |
| + KVMN (In) | 76.28   | 77.10      |
| + TaMM (In) | 76.61   | 77.52      |
| + KVMN (Cross) | 76.25  | 77.06      |
| + TaMM (Cross) | 76.54  | 77.44      |
| + KVMN (Both) | 76.49  | 77.48      |
| + TaMM (Both) | 76.86  | 78.13      |

Table 8: The mean $\mu$ and standard deviation $\sigma$ of accuracy and F1 scores of all models (i.e., the ones using base or large BERT with KVMN or TaMM and different combinations of in-entity and cross-entity dependency information) on the test set of ACE2005 and SemEval for relation extraction.

| Models        | ACE2005 $\mu$ | ACE2005 $\sigma$ | SemEval $\mu$ | SemEval $\sigma$ |
|---------------|---------------|------------------|----------------|------------------|
| BERT-base     | 74.86         | 0.42             | 87.48          | 0.38             |
| + GCN         | 75.15         | 0.31             | 88.02          | 0.16             |
| + GAT         | 75.70         | 0.29             | 88.01          | 0.36             |
| + KVMN (In)   | 76.15         | 0.24             | 88.62          | 0.10             |
| + TaMM (In)   | 76.61         | 0.18             | 88.76          | 0.14             |
| + KVMN (Cross)| 75.99         | 0.42             | 88.43          | 0.16             |
| + TaMM (Cross)| 76.43         | 0.14             | 88.49          | 0.24             |
| + KVMN (Both) | 76.44         | 0.34             | 88.59          | 0.36             |
| + TaMM (Both) | 76.59         | 0.46             | 88.96          | 0.19             |
| BERT-large    | 76.28         | 0.47             | 88.66          | 0.34             |
| + GCN         | 76.29         | 0.46             | 89.15          | 0.26             |
| + GAT         | 76.82         | 0.32             | 89.12          | 0.25             |
| + KVMN (In)   | 76.98         | 0.33             | 89.23          | 0.13             |
| + TaMM (In)   | 77.35         | 0.38             | 89.48          | 0.26             |
| + KVMN (Cross)| 77.06         | 0.13             | 89.19          | 0.17             |
| + TaMM (Cross)| 77.31         | 0.34             | 89.45          | 0.12             |
| + KVMN (Both) | 77.08         | 0.49             | 89.61          | 0.23             |
| + TaMM (Both) | 78.62         | 0.32             | 89.88          | 0.16             |

Appendix B. Mean and Deviation of the Results

In the experiments, we test models with different configurations. For each model, we train it with the best hyper-parameter setting using five different random seeds. We report the mean ($\mu$) and standard deviation ($\sigma$) of the F1 scores on the test set of ACE2005 and SemEval in Table 8.