Relational Triple Extraction: One Step is Enough

Yu-Ming Shang\textsuperscript{1}, Heyan Huang\textsuperscript{1}\textsuperscript{*}, Xin Sun\textsuperscript{1}, Wei Wei\textsuperscript{2} and Xian-Ling Mao\textsuperscript{1}

\textsuperscript{1}School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China
\textsuperscript{2}Huazhong University of Science and Technology, Hu’bei, China
\{ymshang, hhy63, sunxin\}@bit.edu.cn, Weiw@hust.edu.cn, maoxl@bit.edu.cn

Abstract

Extracting relational triples from unstructured text is an essential task in natural language processing and knowledge graph construction. Existing approaches usually contain two fundamental steps: (1) finding the boundary positions of head and tail entities; (2) concatenating specific tokens to form triples. However, nearly all previous methods suffer from the problem of error accumulation, i.e., the boundary recognition error of each entity in step (1) will be accumulated into the final combined triples. To solve the problem, in this paper, we introduce a fresh perspective to revisit the triple extraction task, and propose a simple but effective model, named DirectRel. Specifically, the proposed model first generates candidate entities through enumerating token sequences in a sentence, and then transforms the triple extraction task into a linking problem on a “head → tail” bipartite graph. By doing so, all triples can be directly extracted in only one step. Extensive experimental results on two widely used datasets demonstrate that the proposed model performs better than the state-of-the-art baselines.

1 Introduction

Relational triple extraction, defined as the task of extracting pairs of entities and their relations in the form of (head, relation, tail) or \((h, r, t)\) from unstructured text, is an important task in natural language processing and automatic knowledge graph construction. Traditional pipeline approaches [Zelenko et al., 2003; Chan and Roth, 2011] separate this task into two independent sub-tasks: entity recognition and relation classification while ignoring their intimate connections. Thus, they suffer from the error propagation problem. To tackle this problem, recent studies focus on exploring joint models to extract relational triples in an end-to-end manner.

According to their differences in the extraction procedure, existing joint methods can be broadly divided into three categories: sequence labeling, table filling and text generation. Sequence labeling methods [Zheng et al., 2017; Sun et al., 2019; Yuan et al., 2020; Wei et al., 2020; Zheng et al., 2021; Ren et al., 2022] utilize various tagging sequences to determine the start and end position of entities, sometimes also including relations. Table filling methods [Wang et al., 2020; Yan et al., 2021] construct a table for a sentence and fill each table cell with the tag of the corresponding token-pair. Text generation methods [Zeng et al., 2018; Zeng et al., 2020; Sui et al., 2020; Ye et al., 2021] treat a triple as a token sequence, and employ encoder-decoder architecture to generate triple elements like machine translation.

Although these methods have achieved promising success, most of them suffer from the same problem: error accumulation. Concretely, these methods need to first determine the start and end position of head and tail entities, then splice the corresponding tokens within the entity boundaries to form triples. Unfortunately, the identification of each boundary token may produce errors, which will be accumulated into the predicted triples. As a result, once the recognition of one boundary token fails, the extraction of all triples associated with this token will fail accordingly.

Intuitively, if we can directly extract relational triples from unstructured sentences in a one-step operation without identi-
fying the boundary tokens of entities, the above problem will
be solved. Following this intuition, we revisit the triple
extraction task from a new perspective — bipartite graph link-
ing. As shown in Figure 1, it can be observed that an en-
tity is essentially composed of several consecutive tokens. In
other words, if we exhaustively enumerate token sequences
of a sentence, the result must contain all correct entities.
Thus, the triple (Stephen Chow, Nationality, China) can
be directly identified by predicting whether there is a “link”
Nationality between the two candidate entities “Stephen
Chow” and “China”.

Inspired by the above intuition, in this paper, we propose
a novel relational triple extraction model, named DirectRel,
which is able to directly extract all triples from unstructured
text in one step. Specifically, given a sentence, we first gener-
ate candidate entities by enumerating token sequences during
data pre-processing. Then, we design a link matrix for each
relation to detect whether two candidate entities can form a
valid triple, and transform triple extraction into a relation-
specific bipartite graph linking problem. Obviously, such a
solution would generate redundant negative samples during
the training phase. To address this issue, DirectRel conducts
downsampling on negative entities during training. Extensive
experimental results demonstrate that DirectRel outperforms
the state-of-the-art approaches on two widely used bench-
marks.

In summary, the main contributions of this paper are as
follows:

- We propose a novel perspective to transform the rela-
tional triple extraction task into a bipartite graph linking
problem, which addresses the error accumulation issue
from design.

- As far as we know, the proposed DirectRel is the first
model that is capable of directly extracting all relational
triples from unstructured text with one-step computa-
tional logic.

- We conduct extensive experiments on two widely used
datasets, and the results indicate that our model performs
better than state-of-the-art baselines.

2 Related Work

This paper focuses on the joint extraction of relational triples
from sentences. Related works can be roughly divided into
three categories.

The first category is sequence labeling methods, which
transform the triple extraction task into several interrelated
sequence labeling problems. For example, a classical method
NovelTagging [Zheng et al., 2017] designs a complex tagging
scheme, which contains the information of entity beginning
position, entity end position and relation. Some studies [Sun
et al., 2019; Liu et al., 2020; Yuan et al., 2020] first use se-
quence labeling to identify all entities in a sentence, and then
perform relation detection through various classification net-
works. Recently, Wei et al. [2020] present CasRel, which first
identifies all possible head entities, then for each head entity,
applies relation-specific sequence taggers to identify the cor-
responding tail entities. PRGC [Zheng et al., 2021] designs
a component to predict potential relations, which constrains
the following entity recognition to the predicted relation subset
rather than all relations. BiRTE [Ren et al., 2022] pro-
poses a bidirectional entity extraction framework to consider
head-tail and tail-head extraction order simultaneously.

The second category is table filling methods, which for-
semble the triple extraction task as a table constituted by the
Cartesian product of the input sentence to itself. For example,
GraphRel [Fu et al., 2019] takes the interaction between enti-
ties and relations into account via a relation-weighted Graph
Convolutional Network. TPLinker [Wang et al., 2020] con-
verts triple extraction as a token pair linking problem and in-
troduces a relation-specific handshaking tagging scheme to
align the boundary tokens of entity pairs. PFN [Yan et al.,
2021] utilizes a partition filter network, which generates taskspe-
cific features jointly to model the interactions between enti-
ty recognition and relation classification.

The third category is text generation methods, which treat
a triple as a token sequence and employs the encoder-decoder
framework to generate triple elements like machine transla-
tion. For example, CopyRE [Zeng et al., 2018] generates the
relation followed by its two corresponding entities with a
copy mechanism, but this method can only predict the last
word of an entity. Thus, CopyMTL [Zeng et al., 2020] em-
loys a multi-task learning framework to address the multi-
token entity problem. CGT [Ye et al., 2021] proposes a con-
trastive triple extraction method with a generative transformer
to address the long-term dependence and faithfulness issues.
R-BPtrNet [Chen et al., 2021] designs a binary pointer net-
work to extract explicit triples and implicit triples.

However, nearly all existing methods suffer from the error
accumulation problem due to possible errors in entity bound-
dary identification. Different from previous methods, Direc-
tRel proposed in this paper transforms the triple extraction
task into a bipartite graph linking problem without determin-
ing the boundary tokens of entities. Therefore, our method is
able to directly extract all triples from unstructured sentences
with a one-step linking operation and naturally address the
problem of error accumulation.

3 Method

The overall architecture of the proposed DirectRel is illus-
trated in Figure 2. In the following, we first give the task def-
inition and notations in Section 3.1. Then, the strategies for
candidate entities generation are introduced in Section 3.2.
Finally, Section 3.3 illustrates the details of the bipartite graph
linking based triple extraction.

3.1 Task Definition

The goal of relational triple extraction is to identify all pos-
sible triples in a given sentence. Therefore, the input of our
model is a sentence \( S = \{w_1, w_2, ..., w_L\} \) with \( L \) tokens. Its
output is a set of triples \( T = \{(h, r, t)|h, t \in \hat{E}, r_i \in \mathcal{R}\} \),
where \( \mathcal{R} = \{r_1, r_2, ..., r_K\} \) denotes \( K \) pre-defined relations.
It is worth noting that, \( \hat{E} \) represents the head and tail entities
in triples, not all named entities in the sentence.
3.2 Candidate Entities Generation

During data pre-processing, we enumerate all consecutive token sequences with length less than \( C (C < L) \) in a sentence as its candidate entities. For example, if \( C = 2 \), the candidate entities of the sentence “Beijing is the capital of China” are \( \mathcal{E} = \{ \text{“Beijing”, “Beijing is”, “is”, “is the”, “the”, “the Capital”, “Capital of”, “of”, “of China”, “China”} \} \). Thus, for a sentence with \( L \) tokens, the number of candidate entities \( |\mathcal{E}| \) is:

\[
|\mathcal{E}| = L \times C + \frac{C^2}{2} - \frac{C^2}{2}. \tag{1}
\]

Obviously, such a strategy will bring two disadvantages: First, the training process will bias towards negative triples as they dominate, which will hurt the model’s ability to identify positive triples. Second, since the number of training sentences is large, too many candidate entities will reduce the training efficiency. To address these issues, for a sentence, we randomly sample \( n_{\text{neg}} \) negative entities from \( \mathcal{E} \) to train the model together with all ground truth entities, and the new subset is denoted as \( \bar{\mathcal{E}} \).

3.3 Bipartite Graph Linking

Given a sentence and its candidate entities \( \bar{\mathcal{E}} \), we employ a pre-trained BERT [Devlin et al., 2019] as sentence encoder to obtain the \( d \)-dimensional contextual representation \( h_i \) for each token:

\[
[h_1, h_2, ..., h_L] = BERT([x_1, x_2, ..., x_L]), \tag{2}
\]

where \( x_i \) is the input representation of the \( i \)-th token. It is the summation over the corresponding token embedding and positional embedding.

It’s worth noting that an entity is usually composed of multiple tokens, to facilitate parallel computation, we need to keep the dimension of different entity representations consistent. Therefore, we take the averaged vector between the start token and end token of entity \( e_i \) as its representation:

\[
e_i = \frac{h_{\text{start}} + h_{\text{end}}}{2}. \tag{3}
\]

Then, as shown in Figure 2, we define a directed “head \( \rightarrow \) tail” bipartite graph for triple extraction, which takes the projected entity representations \( E_{\text{head}} = W_h^T E + b_h \) and \( E_{\text{tail}} = W_t^T E + b_t \) as two parts, where \( E \) is the \( d \)-dimensional representations of entities obtained by equation (3); \( W_h, W_t \) are two project matrices from token feature space to \( d_e \)-dimensional head entity space and tail entity space, respectively, allowing the model to identify the head or tail role of each entity; \( b_h, b_t \) is the bias.

Finally, for each relation \( r_k \), we predict the links between every entity pair to determine whether they can form a valid triple:

\[
P_{ij}^k = \sigma(E_{\text{head}}^T U_{ij} E_{\text{tail}}), \tag{4}
\]

where \( \sigma \) is the sigmoid activation function, \( U_{ij}^{d_e \times d_e} \) is a relation-specific link matrix, which models the correlation between two entities with respect to the \( k \)-th relation. The triple \((e_i, r_k, e_j)\) will be treated as correct if the corresponding probability \( P_{ij}^k \) exceeds a certain threshold \( \theta \), or false otherwise. Besides, since the entity spans have been determined in the data pre-processing stage, decoding triples from the output of our model becomes easy and straightforward. That is, for each relation \( r_k \), the predicted triple is \((e_i, \text{span}, r_k, e_j, \text{span})\), if \( P_{ij}^k > \theta \).

Obviously, our method can naturally identify nested entities and overlapping triples\(^4\) [Zeng et al., 2018]. Specifically,

---

\(^4\) More details about different overlapping patterns are shown in Appendix A.
Table 1: Statistics of datasets. $N$ is the number of triples in a sentence, $E$-len denotes the maximum length of entities using byte pair encoding (BPE), which determines the setting of the hyper-parameter $C$ (the length of candidate entities).

| Category | Dataset | Details of Test Set |
|----------|---------|---------------------|
|          | Train   | Valid   | Test   | Relations | Normal | SEO | EPO | HTO | N=1 | N=2 | N=3 | N=4 | N≥5 | Triples | E-len |
| NYT$^*$  | 56,195  | 4,999   | 5,000  | 24       | 3,266   | 1,297| 978 | 45  | 3,244| 1,045| 312 | 291 | 108 | 8,110 | 7     |
| WebNLG$^*$| 5,019   | 500     | 703    | 171      | 245    | 457 | 26  | 84  | 266 | 171  | 131 | 90  | 45  | 1,591 | 6     |
| NYT      | 56,195  | 5,000   | 5,000  | 24       | 3,222   | 1,273| 969 | 117 | 3,240| 1,047| 314 | 290 | 109 | 8,120 | 11    |
| WebNLG   | 5,019   | 500     | 703    | 216      | 239    | 448 | 6   | 85  | 256 | 175  | 138 | 93  | 41  | 1,607 | 39    |

for nested entity recognition, the correct entities must be included in the candidate entities generated by enumeration. For EntityPairOverlap (EPO) case, entity pairs with different relations will be recognized by different relation-specific link matrices. For SingleEntityOverlap (SEO) case, if two triples have the same relation, there will be two edges in the bipartite graph; if two triples have different relations, they will be identified by different link matrices. For HeadTailOverlap (HTO) case, the overlapped entity will appear on both parts of the bipartite graph and can also be easily identified.

3.4 Objective Function
The objective function of DirectRel is defined as:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}| \times K} \sum_{i=1}^{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} (y_i \log(P_{ij}^k) + (1 - y_i) \log(1 - P_{ij}^k)),$$

where $|\mathcal{E}|$ is the number of entities used for training, $K$ denotes the number of pre-defined relations, $y_i$ is the gold label of the triple $(e_i, r_k, e_j)$.

4 Experiments
Our experiments are designed to evaluate the effectiveness of the proposed DirectRel and analyze its properties. In this section, we first introduce the experimental settings. Then, we present the evaluation results and discussion.

4.1 Experimental Settings
Datasets and Evaluation Metrics
We conduct experiments on two widely used relational triple extraction benchmarks: NYT [Riedel et al., 2010] and WebNLG [Gardent et al., 2017].

- **NYT**: The dataset is generated by distant supervision, which automatically aligns relational facts in Freebase with the New York Times (NYT) corpus. It contains 56k training sentences and 5k test sentences.
- **WebNLG**: The dataset is originally developed for Natural Language Generation (NLG) task, which aims to generate corresponding descriptions from given triples. It contains 5k training sentences and 703 test sentences.

Both NYT and WebNLG have two versions: one version only annotates the last word of entities, denoted as NYT$^*$ and WebNLG$^*$; the other version annotates the whole span of entities, denoted as NYT and WebNLG. Table 1 illustrates their detailed statistics. Notably, as our model employs byte pair encoding, entities in NYT$^*$ and WebNLG$^*$ may also contain multiple tokens.

Following previous works [Wei et al., 2020; Zheng et al., 2021; Ren et al., 2022], we adopt standard micro Precision (Prec.), Recall (Rec.) and F1-score (F1) to evaluate the performances. Concretely, a predicted triple $(h, r, t)$ is regarded to be correct only if the head $h$, tail $t$ and their relation $r$ are identical to the ground truth.

Implementation Details
We employ the cased base version$^3$ of BERT as sentence encoder. Therefore, the dimension of token representation $h_i$ is $d = 768$. The dimension of projected entity representations $d_e$ is set to 900. During training, the learning rate is $1e^{-5}$, and the batch size is set to 8 on NYT$^*$ and NYT, 6 on WebNLG$^*$ and WebNLG. The max length of candidate entities $C$ is 9/6/12/21 on NYT$^*$/WebNLG$^*$/NYT/WebNLG respectively. For each sentence, we randomly select $n_{neg} = 100$ negative entities from $\mathcal{E}$ to optimize the objective function of a mini-batch. If there are fewer than 100 candidates in a sentence, all negative entities will be used. During inference, we predict links for all candidate entities and the max length $C$ is 7/6/11/20 on NYT$^*$/WebNLG$^*$/NYT/WebNLG respectively$^4$. All experiments are conducted on a work station with an AMD 7742 2.25G CPU, 256G Memory, and a single RTX 3090 GPU.

Baselines
We compare our method with the following ten baselines: GraphRel [Fu et al., 2019], MHSA [Yuan et al., 2020], RSAN [Liu et al., 2020], CopyMTL [Zeng et al., 2020], CasRel [Wei et al., 2020], TPLinker [Wang et al., 2020], CGT [Ye et al., 2021], PRGC [Zheng et al., 2021], R-BPtrNet [Chen et al., 2021], BiRTE [Ren et al., 2022].

For fair comparison, the reported results for all baselines are directly from the original literature.

$^3$https://huggingface.co/bert-base-cased

$^4$We present the entity length distribution of datasets in Appendix B, as well as our consideration when setting $C$.
TABLE 2: Precision(%) of sentence-level extraction of our proposed DirectRel and baselines.

| Model      | NYT* Prec. | NYT* Rec. | NYT* F1 | WebNLG* Prec. | WebNLG* Rec. | WebNLG* F1 |
|------------|------------|-----------|---------|----------------|-------------|-----------|
|           | 63.9       | 60.0      | 61.9    | 44.7          | 41.1        | 42.9      |
| GraphRel [Fu et al., 2019] | 88.1       | 78.5      | 83.0    | 89.5          | 86.0        | 87.7      |
| CasRel [Wei et al., 2020] | 91.3       | 92.5      | 91.9    | 91.8          | 92.0        | 91.9      |
| TPLinker [Wang et al., 2020] | 94.7       | 84.2      | 89.1    | 92.9          | 75.6        | 83.4      |
| PRGC [Zheng et al., 2021] | 93.3       | 91.9      | 92.6    | 94.0          | 92.1        | 93.0      |
| R-BPtrNet [Chen et al., 2021] | 92.7       | 92.5      | 92.6    | 93.7          | 92.8        | 93.3      |
| BiRTE [Ren et al., 2022] | 92.2       | 93.8      | 93.0    | 93.2          | 94.0        | 93.6      |
| DirectRel | 93.7       | 92.8      | 93.2    | 94.1          | 94.1        | 94.1      |

Table 2: Precision(%) of sentence-level extraction of our proposed DirectRel and baselines. GraphRel, RSAN, MHSA, CopyMTL use LSTM as sentence encoder, while other methods employ a pre-trained BERT to obtain feature representations.

4.2 Results and Analysis

Main Results

In Table 2, we present the comparison results of our DirectRel with ten baselines on two versions of NYT and WebNLG. It can be observed that DirectRel outperforms all the ten baselines and achieves the state-of-the-art performance in terms of F1-score on all datasets. Among the ten baselines, CasRel and TPLinker are the representative methods for combing triples through identifying boundary tokens of head and tail entities. Our DirectRel outperforms CasRel by 3.6 and 2.3 absolute gains in F1-score on NYT* and WebNLG*; and outperforms TPLinker by 1.3, 2.2, 0.9, 3.3 absolute gains in terms of F1-score on NYT*, WebNLG*, NYT and WebNLG respectively. Such results demonstrate that DirectRel achieves the best performance against all baselines, which proves the effectiveness of our strategies of candidate entities generation and negative entities sampling.

Detailed Results on Complex Scenarios

To further explore the capability of our DirectRel in handling complex scenarios, we split the test set of NYT* and WebNLG* by overlapping patterns and triple number, and the detailed extraction results are shown in Table 3. It can be observed that DirectRel obtains the best F1-score on 13 of the 18 subsets, and the second best F1-score on the remaining 5 subsets. Besides, we can also see that DirectRel obtains more performance gains when extracting EPO triples. We attribute the outstanding performance of DirectRel to its two advantages: First, it effectively alleviates the error accumulation problem and ensures the precision of extracted triples. Second, it applies a relation-specific linking between every entity pair, guaranteeing the recall of triple extraction. Overall, the above results adequately prove that our proposed method is more effective and robust than baselines when dealing with complicated scenarios.

Results on Different Sub-tasks

Our DirectRel combines entity recognition and relation classification into a one-step bipartite graph link operation, which can better capture the interactions between the two sub-tasks. Furthermore, the one-step extraction logic protects the model from cascading errors and exposure bias. To verify such prop-
Figure 3: The influence of different $n_{neg}$ in NYT$^*$ and WebNLG$^*$. Training time (ms) means the average time required to train one mini-batch, GPU memory (MB) is the average GPU memory required to train one epoch.

Table 4: Results on triple elements. (h, t) denotes the entity pair and r means the relation.

| Model  | Element | NYT$^*$ | WebNLG$^*$ |
|--------|---------|---------|------------|
|        | Prec.   | Rec.    | F1         | Prec. | Rec. | F1 |
| PRGC   | (h, t)  | 94.0    | 92.3       | 93.1  | 96.0 | 93.4 | 94.7 |
|        | r       | 95.3    | 96.3       | 95.8  | 92.8 | 96.2 | 94.5 |
|        | (h, r, t) | 93.3 | 91.9       | 92.6  | 94.0 | 92.1 | 93.0 |
| DirectRel | (h, t)  | 94.1    | 93.2       | 93.7  | 95.8 | 95.9 | 95.8 |
|        | r       | 97.3    | 96.4       | 96.9  | 96.8 | 96.7 | 96.7 |
|        | (h, r, t) | 93.7 | 92.8       | 93.2  | 94.1 | 94.1 | 94.1 |

Table 5: Distribution of three entity recognition errors on WebNLG.

| Type                  | Distribution |
|-----------------------|--------------|
| Span Splitting Error  | 35.5%        |
| Entity Not Found      | 19.4%        |
| Entity Role Error     | 45.1%        |

Table 5: Distribution of three entity recognition errors on WebNLG.

5 Conclusion

In this paper, we focus on addressing the error accumulation problem in existing relational triple extraction methods, and propose a one-step bipartite graph linking based model, named DirectRel, which is able to directly extract relational triples from unstructured text without specific processes of determining the start and end position of entities. Experimental results on two widely used datasets demonstrate that our model performs better than state-of-the-art baselines, especially for complex scenarios of different overlapping patterns and multiple triples.
6 Acknowledgments

The work is supported by National Key R&D Plan (No. 2020AAA0106600), National Natural Science Foundation of China (No. U21B2009, 62172039, 61732005, 61602197 and L1924068 ), the funds of Beijing Advanced Innovation Center for Language Resources (No. TYZ19005).

References

[Chan and Roth, 2011] Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 551–560, 2011.

[Chen et al., 2021] Yubo Chen, Yunqi Zhang, Changran Hu, and Yongfeng Huang. Jointly extracting explicit and implicit relational triples with reasoning pattern enhanced binary pointer network. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics, pages 5694–5703, 2021.

[Devlin et al., 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, pages 4171–4186, 2019.

[Fu et al., 2019] Tsui-Ju Fu, Peng-Hsuan Li, and Wei-Yun Ma. 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, 2019.

[Gardent et al., 2017] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for NLG micro-planners. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 179–188, 2017.

[Liu et al., 2020] Jie Liu, Shaowei Chen, Bingquan Wang, Jiaxin Zhang, Na Li, and Tong Xu. Attention as relation: Learning supervised multi-head self-attention for relation extraction. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pages 3787–3793, 2020.

[Ren et al., 2022] Feiliang Ren, Longhui Zhang, Xiaofeng Zhao, Shujuan Yin, Shilei Liu, and Bochao Li. A simple but effective bidirectional extraction framework for relational triple extraction. In The 15th ACM International Conference on Web Search and Data Mining, 2022.

[Riedel et al., 2010] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In Machine Learning and Knowledge Discovery in Databases, European Conference, volume 6323, pages 148–163, Springer, 2010.

[Sui et al., 2020] Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. Joint entity and relation extraction with set prediction networks. arXiv preprint arXiv:2011.01675, 2020.

[Sun et al., 2019] Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. Joint type inference on entities and relations via graph convolutional networks. In Proceedings of the 57th Conference of the Association for Computational Linguistics, pages 1361–1370, 2019.

[Wang et al., 2020] Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. TPLinker: Single-stage joint extraction of entities and relations through token pair linking. In Proceedings of the 28th International Conference on Computational Linguistics, pages 1572–1582, 2020.

[Wei et al., 2020] Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. A novel cascade binary tagging framework for relational triple extraction. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1476–1488, 2020.

[Yan et al., 2021] Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. A partition filter network for joint entity and relation extraction. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 185–197, 2021.

[Ye et al., 2021] Hongbin Ye, Ningyu Zhang, Shumin Deng, Mosha Chen, Chuanqi Tan, Fei Huang, and Huajun Chen. Contrastive triple extraction with generative transformer. In Thirty-Fifth AAAI Conference on Artificial Intelligence, pages 14257–14265, 2021.

[Yuan et al., 2020] Yue Yuan, Xiaofei Zhou, Shirui Pan, Qiannan Zhu, Zeliang Song, and Li Guo. A relation-specific attention network for joint entity and relation extraction. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pages 4054–4060, 2020.

[Zelenko et al., 2003] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. J. Mach. Learn. Res., 3:1083–1106, 2003.

[Zeng et al., 2018] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. Extracting relational facts by an end-to-end neural model with copy mechanism. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 506–514, 2018.

[Zeng et al., 2020] Daojian Zeng, Haoran Zhang, and Qianying Liu. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, pages 9507–9514, 2020.

[Zheng et al., 2017] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1227–1236, 2017.

[Zheng et al., 2021] Hengyi Zheng, Rui Wen, Xi Chen, Yifan Yang, Yunyan Zhang, Ziheng Zhang, Ningyu Zhang, Bin Qin, Xu Ming, and Yefeng Zheng. PRGC: potential relation and global correspondence based joint relational triple extraction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, pages 6225–6235, 2021.
Appendix

A Overlapping Patterns

As shown in Figure 4, sentences may contain Entity Pair Overlap (EPO), Single Entity Overlap (SEO) and Head Tail Overlap (HTO) triples. EPO means an entity pair holds multiple relations. SEO denotes two triples share one overlapped entity. HTO indicates that the head entity and tail entity of a triple are partially or completely overlapped.

| Sentences                                                                 | Triples                           |
|--------------------------------------------------------------------------|-----------------------------------|
| **Normal** [Bionico] is a dessert containing sour cream from [Mexico].    | (Bionico, Country, Mexico)        |
| **EPO** [Beijing] is the capital city of [China].                        | (Beijing, Capital of, China)      |
|                                                                         | (Beijing, City of, China)         |
| **SEO** [Obama] was born in [Honolulu], city of [Hawaii].               | (Obama, Place of birth, Honolulu) |
|                                                                         | (Honolulu, City of, Hawaii)       |
| **HTO** [Bruce [Lee]] is a famous martial artist.                        | (Bruce Lee, Family name, Lee)     |

Figure 4: Examples of Normal, EPO, SEO and HTO patterns. The overlapping entities are marked in red.

B Entity Length Distribution

The entity length distribution of NYT*, WebNLG*, NYT and WebNLG are shown in Figure 5. When setting the max length of candidate entities $C$, we mainly consider two factors: (1) The generated candidate entities should contain as many correct entities as possible. (2) The number of negative entities should not be too large.

Therefore, the max length of candidate entities $C$ is set to 9, 6, 12, 21 during training; 7, 6, 12, 21 during validation. Strictly speaking, the distribution of entities in the test set is unknown at the time of model inference. Therefore, during inference, we make adjustments based on the $C$ in the training set, and the final $C$ is set to 7, 6, 11, 20 during testing on NYT*, WebNLG*, NYT and WebNLG respectively.

C Model Efficiency

We randomly select $n_{neg}$ negative entities during training, while predict links for all candidate entities during inference. Therefore, we evaluate the model efficiency with respect to inference time of the token-pair linking method TPLinker in NYT*, WebNLG*, NYT, WebNLG, and the results are shown in Table 6.

| Model     | NYT* | WebNLG* | NYT | WebNLG |
|-----------|------|---------|-----|--------|
| TPLinker  | 46.2 | 40.1    | 45.1| 42.2   |
| DirectRel | 56.2 | 40.3    | 57.9| 38.4   |

Table 6: The inference time (ms) of DirectRel and TPLinker to predict triples of one sentence.

From Table 6 we can see that the inference speed of our DirectRel is competitive with TPLinker. The reason behind this are as follows: (1) Although our model needs to process more samples, its well-designed architecture is much simpler than TPLinker. (2) TPLinker requires a complex decoding algorithm to decode triples from output matrices, while the output of our model is directly the indices of entity spans and relations. Therefore, compared with the number of candidate entities, the model architecture and processing logic are more essential for the inference efficiency, and such number of candidate entities is not a bottleneck for the application of our model.
Figure 5: Entity length distribution of datasets using byte pair encoding.