# A Relation-Attentive 3D Matrix Framework for Relational Triple Extraction

Anonymous ACL submission

## Abstract

Extracting relational triples from unstructured text is crucial for information extraction. Recent methods achieve considerable performance, but due to the insufficient consideration of triple global information, there is an obvious performance gap between triple (E1, R, E2) and E1/R/E2, that is, some extracted entities or relations fail to form a valid relational triple. To break this bottleneck, we propose a relation-attentive 3D matrix framework (RA3D) composed of an encoder module, a fusion module, and a 3D matrix module. Instead of using a 2D table to align the subject and object, we integrate clearly encoded relation information to convert the 2D table into a 3D matrix, so that the entries of the 3D matrix can capture the interaction in subjects, objects, and relations completely. To extract relation and entity information required for the 3D matrix reasonably, we design a transformer-decoder-based fusion module that updates the representation of relations and entities iteratively. Our model achieves state-of-the-art performance with F1 score up to 93.5% and 94.3% on two public datasets and delivers consistent performance gain on complex scenarios of overlapping triples.

## 1 Introduction

Extracting relational facts from natural language text is a well-studied task in information extraction (IE) and a crucial step towards building large structural knowledge bases (KB) (Auer et al., 2007; Bollacker et al., 2008; Dong et al., 2014). A relational fact is represented as a triple that consists of two entities (an entity pair) connected by a semantic relation.

Traditional methods in relational triple extraction take in a pipeline manner (Zelenko et al., 2003; Zhou et al., 2005; Chan and Roth, 2011). It first recognizes all entities in a sentence using a named entity recognizer and then performs relation classification for each entity pair. Such an approach eases the task and makes each component more flexible, but it tends to suffer from the error propagation problem, since the results of entity recognition can affect the performance of relation classification. To tackle this problem, many joint learning models that extract entities and relations in a single model have been proposed. Traditional joint methods (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Ren et al., 2017) are designed with a feature-based structure that needs heavily feature engineering work. With the rapid development of deep learning, many Neural Network-based (NN-based) models (Gupta et al., 2016; Katiyar and Cardie, 2017) achieve state-of-the-art performance. However, extracting overlapping triples remains challenging. Most existing models that handle the overlapping triple problem are multi-stage-based (Zeng et al., 2018; Wei et al., 2020; Zheng et al., 2021). They involve sequential interrelated steps and suffer from the problem of exposure bias. At training time, they predict with the ground truth conditions while at inference they have to make extraction from scratch. This discrepancy leads to error accumulation. To mitigate the issue, single-stage

| | |
|---|---|
| sentence | Alan Bean (of the United States) was a crew member of NASA 's Apollo 12 under the commander David Scott. |
| gold subject | Bean, 12 |
| gold object | 12, States, NASA, Scott |
| gold relation | was a crew member, nationality, operator, commander |
| gold triples | (Bean, was a crew member of, 12 )<br>(Bean, nationality, States)<br>(12, operator, NASA )<br>(12, commander, Scott) |
| predict triples | (Bean, nationality, 12 )<br>(Bean, nationality, NASA )<br>(Bean, operator, Scott )<br>(12, commander, States)<br>(12, was a crew member, States)<br>... |

Figure 1: Scenarios in which there is an obvious performance gap between (E1, R, E2) and E1/R/E2.

frameworks (Wang et al., 2020; Sui et al., 2021) are proposed.

Despite their success, there is still an obvious performance gap between triple (E1, R, E2) and E1/R/E2 where E1 represents the subject entity, E2 represents the object entity and R represents the relation between them as shown in Figure 1. E1 is regarded as correct as long as the subject in the extracted triple is correct, so are E2 and R. To tackle this problem, we design a 3D matrix that is evolved from a 2D table. Casting Named Entity Recognition (NER) and Relation Classification (RC) as a table filling problem (Miwa and Sasaki, 2014; Gupta et al., 2016; Zhang et al., 2017; Wang and Lu, 2020) is a popular idea for a related but different branch of the joint entity and relation extraction that needs to extract entity types but does not focus on the overlapping problem. Recently, many strong baselines (Wang et al., 2020; Zheng et al., 2021) in relational triple extraction borrow the 2D table architecture of table filling and achieve considerable performance. They treat relations as discrete labels, give relation-specific 2D table representations, or do not consider relation information in the 2D table. Since none of them integrate well-encoded relation information to make the best of triple global information, the performance gap between (E1, R, E2) and E1/R/E2 is still obvious.

In this paper, we propose a relation-attentive 3D matrix framework (RA3D) to narrow the performance gap between (E1, R, E2) and E1/R/E2. Firstly, we design two different encoders – a sentence encoder and a relation encoder to capture the two different types of information. Then, we propose a fusion module that enhances the sentence and relation representation capabilities to make the information conveyed into the 3D matrix more reasonable. In the fusion module, we leverage the transformer-decoder to query the related information between the sentences and relations and further design a similarity gate to update the representations accurately with the related information queried. Finally, a relational triple 3D matrix is formed where each entry captures the interaction among a subject, a relation, and an object. The representation of all possible subjects and objects is the sentence output of the fusion module, and the representation of relations is the relation output of the fusion module. This work has the following main contributions:

1. We propose a relation-attentive 3D matrix framework. It interacts the well-encoded relation information with all possible subjects and objects in a 3D matrix to narrow the performance gap between (E1, R, E2) and E1/R/E2.

2. To make the representations more conducive to the relational triple extraction task, we design a transformer-decoder-based fusion module that updates the sentence and relation representations iteratively.

3. Extensive experiments on two public datasets show that the proposed framework outperforms state-of-the-art methods, achieving 1.4 and 1.5 absolute gain in F1-score on the two datasets respectively. In addition, the gap between (E1, R, E2) and E1/R/E2 decreases by 1.5 and 0.8 on the two datasets.

## 2 Related Work

Early works (Mintz et al., 2009; Gormley et al., 2015) usually extract relational triples in two separate steps: NER and RC. By employing NER to give sentences with annotated entities, RC can identify the relational facts between the annotated entities. However, such a pipeline manner approach suffers from error propagation problems and neglects the relevance of entity extraction and relation prediction. To tackle this problem, joint learning frameworks which extract entities together with relations have been built. Some of the frameworks are feature-based models (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Ren et al., 2017), and, more recently, others are NN-based models (Gupta et al., 2016; Katiyar and Cardie, 2017; Zheng et al., 2017; Zeng et al., 2018; Fu et al., 2019) which achieve considerable success.

However, early NN-based methods (Miwa and Bansal, 2016) achieve joint learning of entities and relations only through parameter sharing but not joint decoding. They still have separate components for NER and RC subtasks. Different from them, Zheng et al. (2017) introduce a novel tagging scheme to extract entities and their relations achieving joint decoding without identifying entities and relations separately. They show promising results but completely give up overlapping triples.

Most existing models in handling overlapping cases- EntityPairOverlap (EPO) and SingleEntiy-Overlap (SEO) are multi-stage-based models that can be categorized into two classes: decoder-based and decomposition-based. Decoder-based

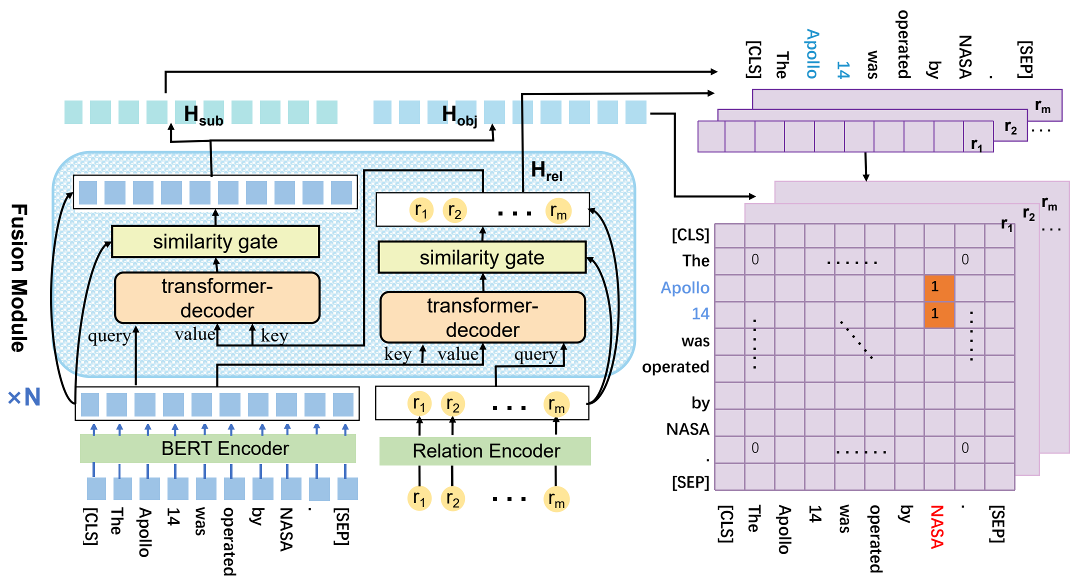

Figure 2: An overview of the proposed RA3D framework. BERT Encoder and Relation Encoder are used to learn representations of the source sentence and relations. Then, a fusion module containing $N$ fusion layers is constructed for more reasonable relation representation $\boldsymbol{H}_{rel}$ and sentence representation $\boldsymbol{H}_{sub}$ and $\boldsymbol{H}_{obj}$. Finally, a 3D matrix is formed by integrating the $\boldsymbol{H}_{obj}$ into the 2D table contains the information of $\boldsymbol{H}_{sub}$ and $\boldsymbol{H}_{rel}$. The orange blocks tagged 1 reflect that the relational triple (Apollo 14, operator, NASA) is extracted.

models use encoder-decoder architecture where the decoder extracts one word or one tuple at a time (Zeng et al., 2018; Nayak and Ng, 2020). Decomposition-based models have an extraction order of triple elements (Wei et al., 2020; Zheng et al., 2021), for example, Wei et al. (2020) first distinguish all the candidate subject entities that may be involved with target triples, then label corresponding object entities and relations for each extracted subject. Although these multi-stage-based methods have achieved reasonable performance, they all suffer from exposure bias. Wang et al. (2020) employ a token pair linking scheme aligning subjects with objects under each relation type in one stage to solve the problem.

Miwa and Sasaki (2014) tackle joint NER and RE as from a table filling perspective, where the entry at row i and column j of the table corresponds to the pair of i-th and j-th word of the input sentence. The diagonal of the table is filled with the entity tags and the rest with the relation tags indicating possible relations between word pairs. Many similar methods (Gupta et al., 2016; Luan et al., 2019; Wang and Lu, 2020) are proposed to fill the table. Recent works (Wang et al., 2020; Zheng et al., 2021) in relational triple extraction borrow the 2D table architecture of table filling. The state-of-the-art method named PRGC (Zheng et al., 2021) designs a global correspondence component to align the subject and object into a triple which is a 2D architecture. However, none of them fuse clearly encoded relation information to the 2D table to consider triple global information more comprehensively.

## 3 Method

In this section, we describe the detail of RA3D framework. An overview illustration of RA3D is shown in Figure 2. The model is composed of the following three modules: an encoder module, a fusion module, and a 3D matrix module.

### 3.1 Encoder Module

#### 3.1.1 Sentence Encoder

BERT is a multi-layer bidirectional Transformer structure model designed to learn deep representations, which has been proven to be effective on several tasks. We employ a pre-trained BERT (Devlin et al., 2018) to encode the context information. The output of sentence encoder is $\boldsymbol{H}_n \in \mathbb{R}^{n\times h}$, where $n$ is the sentence length, and $h$ is the size of hidden state.

#### 3.1.2 Relation Encoder

We design an independent relation encoder which is defined as follows:

$$\boldsymbol{H}_m = \boldsymbol{W}_r\boldsymbol{E}([r_1, r_2, ..., r_m]) + \boldsymbol{b}_r \qquad (1)$$

where $r_i$ is the one-hot vectors of relation indices in the predefined relations, and $m$ is the number of predefined relations. $\boldsymbol{E}$ is the relation embedding matrix, and $\boldsymbol{W}_r$ and $\boldsymbol{b}_r$ are trainable parameters. $\boldsymbol{H}_m \in \mathbb{R}^{m \times h}$ is the output of relation encoder.

### 3.2 Fusion Module

The purpose of the fusion module is to further enhance the expression ability by making the sentence representation and relation representation contain information related to each other reasonably. The fusion module is composed of multiple fusion layers. Each fusion layer updates the relation representation first and the sentence representation second. When we update the relation representation, we treat each relation as a query and generate its related semantic information from the sentence based on the transformer-decoder. We then design a similarity gate to fuse the relation-related information into relation representation. Similarly, we use transformer-decoder and similarity gate to update sentence representation by treating each word in the sentence as a query and relations as providers of key-value pairs.

#### 3.2.1 Transformer-decoder Based Related Information Representation

We generate the related semantic representation between the sentence and relations based on transformer-decoder (Vaswani et al., 2017) which has three sub-layers. Take updating sentence representations as an example. The first is a multi-head self-attention mechanism to model the relationship between word queries, the second is a multi-head cross attention mechanism to map a word query and a set of key-value relation pairs to an output, and the third is a position-wise fully connected feed-forward network. We employ residual connections around each of the sub-layers, followed by layer normalization.

There are two points worth noting. Firstly, since the proposed transformer-decoder directly outputs the final relation representation or sentence representation in one shot instead of one by one, our decoder is non-autoregressive. The autoregressive decoder needs to use no casual mask to prevent positions from attending to subsequent positions. Without the constraint of an autoregressive factorization of the output distribution, we use the unmasked self-attention instead, which is the same as Gu et al. (2018). Secondly, the relations are independent of each other, so there is no need to model the relationship between relation queries. We delete the first sub-layer of the transformer-decoder when updating relation representations.

#### 3.2.2 Similarity Gate Fusion Mechanism

To integrate the query-related information into the query more accurately, we design a similarity gate that can maintain the non-linear capability and prevent attending to irrelevant information. We calculate the semantic similarity $sim_i$ between each query $\boldsymbol{h}_{q_i}$ and its related information $\boldsymbol{h}_{q-related_i}$ by the concatenation, linear, and $Sigmoid$ normalization operation where $i \in [1, Q]$, and $Q$ is the number of queries. If $sim_i$ is less than the set threshold $\alpha$, we assign 0 to the attention score of the corresponding query-related information, and then the query-related information will be excluded in subsequent fusion. If $sim_i$ is greater than the set threshold $\alpha$, we assign the query-related information with the attention score of $sim_i$ to maintain the non-linear capability. We define the above similarity gate fusion mechanism as follows:

$$\begin{aligned} sim_i &= Sigmoid(\boldsymbol{W}_{sim}[\boldsymbol{h}_{q_i}; \boldsymbol{h}_{q-related_i}] + \boldsymbol{b}_{sim}) \\ g_i &= \begin{cases} sim_i, & sim_i > \alpha \\ 0, & sim_i \le \alpha \end{cases} \qquad (2) \\ \boldsymbol{h}'_{q_i} &= g_i \cdot (\boldsymbol{W}_g \boldsymbol{h}_{q-related_i} + \boldsymbol{b}_g) + (1 - g_i) \cdot \boldsymbol{h}_{q_i} \end{aligned}$$

where $\boldsymbol{W}_{sim}$, $\boldsymbol{b}_{sim}$, $\boldsymbol{W}_g$, and $\boldsymbol{b}_g$ are trainable weights, $g$ is the similarity gate, $\cdot$ is element-wise production, and $\boldsymbol{h}'_{q_i}$ is the final output.

#### 3.2.3 Relation-sentence Representation Iterative Fusion

In this section, we introduce the overall architecture of the proposed fusion module. To simplify, we define the above transformer-decoder and similarity gate formulas as follows:

$$\widetilde{\boldsymbol{h}}_{q_i} = \boldsymbol{TD} - \boldsymbol{SG}(\boldsymbol{h}_{q_i}, \boldsymbol{H}_{kv}) \qquad (3)$$

where $\boldsymbol{H}_{kv} = \{\boldsymbol{h}_{kv_j}\}_{j \in [1,K]}$ is the set of all vectors that the query $\boldsymbol{h}_{q_i}$ needs to calculate similarity with. $\widetilde{\boldsymbol{h}}_{q_i}$ is the updated query $\boldsymbol{h}_{q_i}$ representation.

In each fusion layer, we obtain the new relation representation first, and then we update the sentence representation according to the new relation representation. We add a residual connection to avoid gradient vanishing during training after each update process. The $l$-th fusion layer can be represented as follows:

$$\begin{aligned}
\widetilde{\boldsymbol{h}}_{m_i}^{l+1} &= \boldsymbol{TD}-\boldsymbol{SG}(\boldsymbol{h}_{m_i}^{l}, \boldsymbol{H}_{n}^{l}) \\
\boldsymbol{h}_{m_i}^{l+1} &= \widetilde{\boldsymbol{h}}_{m_i}^{l+1} + \boldsymbol{h}_{m_i}^{l} \\
\widetilde{\boldsymbol{h}}_{n_j}^{l+1} &= \boldsymbol{TD}-\boldsymbol{SG}(\boldsymbol{h}_{n_j}^{l}, \boldsymbol{H}_{m}^{l+1}) \\
\boldsymbol{h}_{n_j}^{l+1} &= \widetilde{\boldsymbol{h}}_{n_j}^{l+1} + \boldsymbol{h}_{n_j}^{l}
\end{aligned} \quad (4)$$

where $\boldsymbol{H}_m^l = \{\boldsymbol{h}_{m_i}^l\}_{i\in[1,m]}$ and $\boldsymbol{H}_n^l = \left\{\boldsymbol{h}_{n_j}^l\right\}_{j\in[1,n]}$ are the relations and sentence representation of $l$-th fusion layer. $\boldsymbol{h}_{m_i}^{l+1}$, and $\boldsymbol{h}_{n_j}^{l+1}$ are the output relation and word representation of $l$-th fusion layer.

## 3.3 3D Matrix Module

A 3D matrix triple extraction module is developed to integrate relation information and sentence information to a novel 3D matrix structure and then extract relational triples from the 3D matrix.

In the first stage, a 2D table is formed where each entry captures the interaction between a subject and a relation. Next, a 3D matrix is identified by calculating the interaction between the subject-relation 2D table and each object. Finally, we adopt a binary classifier to detect the triples by assigning each entry a binary tag (0/1) that indicates whether the current entry containing the information of a subject, an object, and a relation corresponds to a triple in the sentence. We define the input vector:

$$\begin{aligned}
\boldsymbol{H}_{sub} &= \boldsymbol{H}_{obj} = \boldsymbol{H}_n \\
\boldsymbol{H}_{rel} &= \boldsymbol{H}_m
\end{aligned} \quad (5)$$

where $\boldsymbol{H}_{sub}$ and $\boldsymbol{H}_{obj}$ are set to the output sentence representation of the fusion module represented as $\boldsymbol{H}_n$, and $\boldsymbol{H}_{rel}$ is set to the relation representation of the fusion module represented as $\boldsymbol{H}_m$. Before integrating information to 3D matrix, $\boldsymbol{H}_{sub}$, $\boldsymbol{H}_{obj}$, and $\boldsymbol{H}_{rel}$ are transformed into $\boldsymbol{H}_{sub}^{'}$, $\boldsymbol{H}_{obj}^{'}$, and $\boldsymbol{H}_{rel}^{'}$ which are prepared for the later interaction. The detailed operations are as follows:

$$\begin{aligned}
\boldsymbol{H}_{sub}^{'} &= SplitHead(\boldsymbol{W}_{sub}\boldsymbol{H}_{sub} + \boldsymbol{b}_{sub}) \\
\boldsymbol{H}_{obj}^{'} &= SplitHead(\boldsymbol{W}_{obj}\boldsymbol{H}_{obj} + \boldsymbol{b}_{obj}) \\
\boldsymbol{H}_{rel}^{'} &= Expand(\boldsymbol{H}_{rel})
\end{aligned} \quad (6)$$

where $\boldsymbol{W}_{sub} \in \mathbb{R}^{h\times(h\times m)}$, $\boldsymbol{W}_{obj} \in \mathbb{R}^{h\times(h\times m)}$, $\boldsymbol{b}_{sub}$, and $\boldsymbol{b}_{obj}$ are trainable parameters. $m$ represents the number of predefined relation types in the dataset. We denote a reshape operation as $SplitHead(\cdot)$ in which the embedding vectors of length $h \times m$ are split into embeddings for each relation. Since the length $n$ of each sentence is different, we use $Expand$ to expand $\boldsymbol{H}_{rel}$ $n$ times instead of operating it like $\boldsymbol{H}_{sub}$ and $\boldsymbol{H}_{obj}$. $\boldsymbol{H}_{sub}^{'}$, $\boldsymbol{H}_{obj}^{'}$, and $\boldsymbol{H}_{rel}^{'} \in \mathbb{R}^{(n\times m\times h)}$ are uesd to calculating the interaction as follows:

$$\boldsymbol{H}_{subrel} = \boldsymbol{W}_{subrel}[\boldsymbol{H}_{sub}^{'}; \boldsymbol{H}_{rel}^{'}] \quad (7)$$

where $\boldsymbol{H}_{subrel} \in \mathbb{R}^{(n\times m\times h)}$ is a 2D table capturing the interaction between subjects and relations by the concatenation and linear operation. $\boldsymbol{W}_{subrel} \in \mathbb{R}^{(2h\times h)}$ is learnable parameters. To construct the 3D matrix, we compute the dot products between $\boldsymbol{H}_{subrel}$ and all possible objects, and apply a $Sigmoid$ function to normalize probability matrix to range (0, 1). We operate as follows:

$$\boldsymbol{H}_{subrelobj} = Sigmoid(\boldsymbol{H}_{subrel} \cdot \boldsymbol{H}_{obj}^{'}) \quad (8)$$

$\boldsymbol{H}_{subrelobj} \in \mathbb{R}^{(n\times m\times n)}$ is an asymmetric 3D matrix, because $(e1, r, e2)$ and $(e2, r, e1)$ are not the same triple. Each entry of $\boldsymbol{H}_{subrelobj}$ can be treated as the probability score of the existence of a triple. If the probability of a triple is bigger than the threshold we set, the triple is extracted.

Notably, most previous works that just pay attention to the start/end position of an entity lead to poor generalization, and others that tag each token with BIO (i.e., Begin, Inside, and Outside) lead to more parameters. However, our approach identifies the entities by collecting consecutive extracted token pairs to capture global representations of the entities. At the same time, we use the $Sigmoid$ function as a binary tagger which does not increase as many parameters as BIO. Further, compared with previous works that need to answer which is the relationship between the entity pairs, the binary tagging scheme only needs to answer whether or not the entity pairs have this relationship, which overwhelmingly reduces the difficulty of the triple extraction problem.

## 3.4 Bias Objective Function

The 3D matrix optimizes the following likelihood function to identify the triple (s, r, o) given a sentence representation x:

$$\begin{aligned}
&p_\theta((s,r,o)|x) \quad (9) \\
&= \prod_{s,o=1}^{n}\prod_{r=1}^{m}(p_{sro})^{\boldsymbol{I}\{y_{sro}=1\}}(1-p_{sro})^{\boldsymbol{I}\{y_{sro}=0\}}
\end{aligned}$$

where $n$ is the length of the sentence, and $m$ is the number of predefined relation types. $\boldsymbol{I}\{z\}=\beta$ if z is true and 0 otherwise. $y_{sro}$ is the true binary tag of triple (s, r, o). $p_{sro}$ is the normalized probabilities of tags defined in Formula 8. $\beta$ is the bias weight. The larger the $\beta$ is, the greater the influence of gold labels has. If $\beta$ is big, the model tends to extract more triples, which leads to a decrease in precision value and an increase in recall value. To ease this issue, a suitable threshold $\lambda$ that aims to balance the precision and recall is important. $\beta$ and threshold $\lambda$ are two hyperparameters that are better to tune together. Formally, given annotated sentence $x_j$ from the training set $D$ and a set of potentially overlapping triples $T_j = \{(s, r, o)\}$ in $x_j$, we aim to maximize the data likelihood:

$$L = max \sum_{j=1}^{|D|} \sum_{(s,r,o)\in T_j} logp_{\theta}((s, r, o)|x_j) \quad (10)$$

## 4 Experiments

### 4.1 Datasets and Evaluation Metrics

To evaluate the performance of our methods, we use the public dataset NYT (Riedel et al., 2010) and WebNLG (Gardent et al., 2017), both of which have two versions, respectively. We denote the different versions as NYT*, NYT and WebNLG*, WebNLG. NYT* and WebNLG* annotate the last word of the entities, while NYT and WebNLG annotate the whole entity span. NYT* and NYT datasets are produced by a distant supervision method. They contain 1.18M sentences sampled from 294k 1987-2007 New York Times news articles and have 24 predefined relation types. WebNLG* and WebNLG datasets are adopted from Natural Language Generation (NLG) task for relational triple extraction. WebNLG* dataset contains 171 predefined relation types, while WebNLG contains 216. All datasets contain sentences with multiple relational triples, so they are suitable to be the testbed for evaluating models on extracting overlapping relational triples.

Following previous work (Fu et al., 2019; Wang et al., 2020; Zheng et al., 2021), we adopt the standard Precision (Prec.), Recall (Rec.), and F1-score to evaluate the results. In our experiments, to keep in line with previous works, we use Partial Match for NYT* and WebNLG*, which means the predicted triplets are seen as correct if and only if the relation and the heads of the two corresponding entities are all correct. For NYT and WebNLG, we use Exact Match, which means that the whole spans of subject and object are needed to be matched. The implementation details are shown in Appendix A.

### 4.2 Experimental Result

We compare our RA3D model with several strong baseline models, including NovelTagging (Zheng et al., 2017), CopyR (Zeng et al., 2018), GraphRel (Fu et al., 2019), WDec (Nayak and Ng, 2020), RSAN (Yuan et al., 2020), CasRel (Wei et al., 2020), TPLinker (Wang et al., 2020), SPN (Sui et al., 2021), and PRGC (Zheng et al., 2021). The reported results for the above baselines are directly copied from the original published literature. Our re-implementation results are obtained by the official implementation with default configuration.

#### 4.2.1 Main Results

Table 1 shows the results of our model against other baseline methods on all datasets. Our model overwhelmingly outperforms all the baselines in terms of almost all three evaluation metrics and achieves the state-of-the-art performance in the public datasets. There is a performance gap between the dataset only annotating the last word and the one that annotates the whole span, because identifying the last word of an entity is easier than identifying the whole span.

It is important to note that we design a discard mechanism that discards the triples with incomplete subjects or objects to increase the precision of our model. Before inputting the sentence into BERT, we tokenize the words in the sentence with a designed tokenizer which adds an 'Unused' token after the word tokens. When extracting the subject or object by collecting consecutive extracted token pairs, if the start token is not the next token of the 'Unused' token or the end token is not the previous token of the 'Unused' token, the subject or object will be regarded as incomplete. In this way, our model significantly outperforms the strongest baseline by 1.9 and 1.6 absolute gain in precision on public datasets NYT*, WebNLG* respectively.

#### 4.2.2 Detailed Results on Sentences with Different Overlapping Pattern

To verify the capability of our models in handling the overlapping problem, we conduct further experiments on NYT* dataset and WebNLG* dataset. The detailed results on three different overlapping patterns are presented in Table 2.

| Method | NYT* | | | WebNLG* | | | NYT | | | WebNLG | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Prec.* | *Rec.* | *F1* | *Prec.* | *Rec.* | *F1* | *Prec.* | *Rec.* | *F1* | *Prec.* | *Rec.* | *F1* |
| NovelTagging | - | - | - | - | - | - | 32.8 | 30.6 | 31.7 | 52.5 | 19.3 | 28.3 |
| CopyRE | 61.0 | 56.6 | 58.7 | 37.7 | 36.4 | 37.1 | - | - | - | - | - | - |
| GraphRel | 63.9 | 60.0 | 61.9 | 44.7 | 41.1 | 42.9 | - | – | - | - | - | - |
| WDec | 94.5 | 76.2 | 84.4 | | | | | | | | | |
| RSAN | - | - | - | - | - | - | 85.7 | 83.6 | 84.6 | 80.5 | 83.8 | 82.1 |
| CasRel | 89.7 | 89.5 | 89.6 | 93.4 | 90.1 | 91.8 | - | - | - | - | - | - |
| TPLinker | 91.3 | 92.5 | 91.9 | 91.8 | 92.0 | 91.9 | 91.4 | **92.6** | 92.0 | 88.9 | 84.5 | 86.7 |
| SPN | 93.3 | 91.7 | 92.5 | 93.1 | 93.6 | 93.4 | 92.5 | 92.2 | 92.3 | - | - | - |
| PRGC | 93.3 | 91.9 | 92.6 | 94.0 | 92.1 | 93.0 | 93.5 | 91.9 | 92.7 | 89.9 | 87.2 | 88.5 |
| SPN* | 92.6 | 91.6 | 92.1 | 92.4 | 93.2 | 92.8 | 92.9 | 91.7 | 92.3 | 84.5 | 82.3 | 83.4 |
| PRGC* | 92.0 | 89.7 | 90.8 | 92.8 | 92.4 | 92.6 | 92.5 | 89.6 | 91.0 | 90.4 | 87.2 | 88.8 |
| Ours | **94.5** | **92.5** | **93.5** | **95.0** | **93.6** | **94.3** | **93.9** | 91.9 | **92.9** | **90.6** | **89.2** | **89.9** |

Table 1: Comparison of the proposed RA3D method with the prior works. Bold marks the highest score. Our re-implementation is marked by *.

| Method | NYT* | | | WebNLG* | | |
|---|---|---|---|---|---|---|
| | *Normal* | *SEO* | *EPO* | *Normal* | *SEO* | *EPO* |
| CopyR | 66 | 48.6 | 55 | 59.2 | 33 | 36.6 |
| GraphRel | 69.6 | 51.2 | 58.2 | 65.8 | 38.3 | 40.6 |
| CasRel | 87.3 | 91.4 | 92.0 | 89.4 | 92.2 | 94.7 |
| TPLinker | 90.1 | 93.4 | 94.0 | 90.1 | 93.4 | 94.0 |
| SPN* | 89.8 | 93.9 | 94.5 | 89.1 | 93.5 | 94.9 |
| PRGC* | 88.4 | 92.7 | 93.4 | 88.4 | 93.4 | **95.4** |
| Ours | **92.4** | **94.7** | **94.7** | **92.4** | **94.6** | 95.3 |

Table 2: F1-score of extracting relational triples from sentences with different overlapping pattern.

| | Method | N=1 | N=2 | N=3 | N=4 | N >5 |
|---|---|---|---|---|---|---|
| NYT* | CopyR | 67.1 | 58.6 | 52.0 | 53.6 | 30.0 |
| | GraphRel | 71.0 | 61.5 | 57.4 | 55.1 | 41.1 |
| | CasRel | 88.2 | 90.3 | 91.9 | 94.2 | 83.7 |
| | TPLinker | 90.0 | 92.8 | 93.1 | **96.1** | 90.0 |
| | SPN* | 89.8 | 93.5 | **94.3** | 95.6 | 90.2 |
| | PRGC* | 89.0 | 91.9 | 92.5 | 95.6 | 86.2 |
| | RA3D | **92.3** | **93.8** | 93.5 | 96.0 | **94.1** |
| WebNLG* | CopyR | 59.2 | 42.5 | 31.7 | 24.2 | 30.0 |
| | GraphRel | 66.0 | 48.3 | 37.0 | 32.1 | 32.1 |
| | CasRel | 89.3 | 90.8 | 94.2 | 92.4 | 90.9 |
| | TPLinker | 88.0 | 90.1 | 94.6 | 93.3 | 91.6 |
| | SPN* | 88.6 | 90.6 | **96.3** | 94.2 | 93.3 |
| | PRGC* | 88.4 | 91.9 | 94.0 | 94.8 | 92.9 |
| | RA3D | **91.8** | **93.2** | 95.6 | **95.1** | **95.1** |

Table 3: F1-score of extracting relational triples from sentences with different number of triples.

### 4.2.3 Detailed Results on Sentences with Different Number of Triples

We compare our model's capability in extracting relations from sentences that contain a different number of triplets. We split the sentences into five classes and the detailed results are presented in Table 3. Our model attains consistently strong performance over almost all five classes again.

Our model suffers the least from the increasing complexity of the input sentence. Especially for the most difficult class (N$\geq$5), our model outperforms the strongest baseline by 3.9% and 1.8% improvements on NYT* and WebNLG* datasets. RA3D also presents a significant improvement on the easiest sentences, ones with only one triple, outperforming the strongest baseline by 2.3 and 2.5 absolute gain in F1-score on two public datasets. Experimental results demonstrate the powerful ability of our model in extracting multiple relational triples from both complicated sentences and simple sentences.

## 5 Analysis and Discussion

### 5.1 Ablation Study

| | Model | Prec. | Rec. | F1 |
|---|---|---|---|---|
| WebNLG* | - sentence update | 93.8 | 93.2 | 93.5 |
| | - relation update | 94.4 | 92.9 | 93.6 |
| | -similarity gate mechanism | 94.6 | 93.4 | 94.0 |
| | -bias objective function | 94.2 | 93.4 | 93.8 |
| NYT* | - sentence update | 93.9 | 92.1 | 93.0 |
| | - relation update | 94.1 | 92.5 | 93.3 |
| | -similarity gate mechanism | 94.3 | 92.6 | 93.4 |
| | -bias objective function | 93.9 | 92.8 | 93.3 |

Table 4: Ablation study of RA3D (%). '-' means we remove or change the module from the original RA3D.

In this section, we conduct ablation experiments to demonstrate the effectiveness of each module component in RA3D with results reported in Table 4. We study the impact of sentence representation update and relation representation update. We also replace the similarity gate with the gate used in Zhao et al. (2021). Ours without bias objective function is the special case where parameter $\beta$ is set to 1 and threshold $\lambda$ is set to 0.5.

## 5.2 The Number of Fusion Layers

To confirm the number of the fusion module layers, we study the results of using different numbers of fusion layers on NYT* and WebNLG*. Table 5 presents the results. We can observe that RA3D has the best result for l = 2.

| Number | NYT* | | | WebNLG* | | |
|---|---|---|---|---|---|---|
| | *Prec.* | *Rec.* | *F1* | *Prec.* | *Rec.* | *F1* |
| l=0 | 94.1 | 92.2 | 93.1 | 94.9 | 93.2 | 94.0 |
| l=1 | 94.3 | **92.7** | **93.5** | 94.6 | **93.6** | 94.1 |
| l=2 | **94.5** | 92.5 | **93.5** | **95.0** | **93.6** | **94.3** |
| l=3 | 93.9 | 92.6 | 93.2 | 94.7 | 92.7 | 93.7 |

Table 5: F1-score of different number of fusion layers.

## 5.3 Error Analysis

| Element | NYT* | | | WebNLG* | | |
|---|---|---|---|---|---|---|
| | *CasRel* | *PRGC** | *RA3D* | *CasRel* | *PRGC** | *RA3D* |
| E1 | 93.5 | 94.0 | 95.4 | 95.7 | 97.3 | 97.5 |
| E2 | 93.5 | 94.2 | 95.4 | 95.3 | 96.1 | 97.0 |
| R | 94.9 | 95.1 | 96.0 | 94.0 | 94.8 | 95.9 |
| (E1, R) | 92.2 | 92.9 | 94.6 | 92.5 | 93.5 | 95.0 |
| (R, E2) | 92.2 | 92.7 | 94.5 | 93.2 | 93.8 | 95.1 |
| (E1, E2) | 89.7 | 91.2 | 93.7 | 93.5 | 94.7 | 95.8 |
| (E1, E2, R) | 89.6 | 90.8 | 93.5 | 91.8 | 92.6 | 94.3 |
| gap | 4.4 | 3.6 | 2.1 | 3.2 | 3.5 | 2.5 |
| efficiency | 95.3% | 96.2% | 97.8% | 96.6% | 96.4% | 97.4% |

Table 6: F1-score of different relational triple elements.

In order to verify whether our model has the ability to narrow the performance gap between (E1, R, E2) and E1/R/E2, we analyze the performance on predicting different elements of the triple (E1, R, E2) where E1 represents the subject entity, E2 represents the object entity and R represents the relation between them. An element like (E1, R) is regarded as correct only if the subject and the relation in the predicted triple (E1, R, E2) are both correct, regardless of the correctness of the predicted object. Similarly, we say an instance of E1 is correct as long as the subject in the extracted triple is correct, so sre E2 and R. The **gap** in table 6 is the difference between (E1, R, E2) and the average of E1, E2, and R. The **efficiency** is the percentage value of (E1, R, E2) divided by the average of E1, E2, and R.

Table 6 shows the results on different relational triple elements. For both datasets, the performance gap between RA3D and other models on E1, E2, and R shows our advantages in entity recognition and relation prediction. Compared with CasRel and PRGC*, our model narrows the gap between (E1, R, E2) and E1/R/E2 and achieves encouraging 1.5% and 0.8% declines on NYT* and WebNLG*. As for conversion efficiency. we gain considerable 1.6% and 0.8% improvements on the two datasets respectively. The results indicate that our model has more advantages in identifying the relationship between triple elements than other works.

## 5.4 Model Efficiency

| Epoch | CasRel | TPLinker | PRGC | RA3D |
|---|---|---|---|---|
| 12 | 0.0 | 77.1 | 86.9 | **92.7** |
| 24 | - | - | 91.3 | **94.4** |

Table 7: F1-score at epoch 12 and 24 on the WebNLG* validation set of different methods.

As shown in Table 7, we have a convergence rate superiority. For WebNLG* dataset, we achieve a 92.7% performance at epoch 12 and 94.4% at epoch 24. The result outperforms the PRGC which advantages in convergence rate by 5.8% and 3.1% absolute gain in F1-score at epoch 12 and epoch 24. Results of CasRel, TPLinker, and PRGC are directly taken from Zheng et al. (2021) unless specified. The computation complexity of our model is $O(kn^2)$ for NYT* and $O(n^3)$ for WebNLG* which is similar to TPlinker (Wang et al., 2020). Our method can not only achieve good results but also has obvious advantages in convergence rate, which makes the high complexity acceptable.

## 6 Conclusion

In this paper, we introduce a novel relation-attentive 3D matrix framework (RA3D) for relational triple extraction. It learns an independent relation encoder in addition to the sentence encoder and integrates the clearly encoded relation information to the 3D matrix. We also introduce a new transformer-decoder and similarity gate-based fusion module where explicit interactions exist between the two encoders to enhance their representation abilities. Experimental results show that our model overwhelmingly outperforms state-of-the-art baselines over different scenarios, especially on narrowing the gap between (E1, R, E2) and E1/R/E2.

## References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

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

Yee Seng Chan and Dan Roth. 2011. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

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.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 541–550.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. *arXiv preprint arXiv:1904.03296*.

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8528–8535.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2021. Joint entity and relation extraction with set prediction networks.

In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7072–7079.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. *arXiv preprint arXiv:2010.03851*.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020. 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.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. 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.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Coling 2010: Posters*, pages 1399–1407.

Yue Yuan, Xiaofei Zhou, Shirui Pan, Qiannan Zhu, Zeliang Song, and Li Guo. 2020. A relation-specific attention network for joint entity and relation extraction. In *IJCAI*, pages 4054–4060.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. 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 (Volume 1: Long Papers)*, pages 506–514.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740.

Kang Zhao, Hua Xu, Yue Cheng, Xiaoteng Li, and Kai Gao. 2021. Representation iterative fusion based on heterogeneous graph neural network for joint entity and relation extraction. *Knowledge-Based Systems*, 219:106888.

Hengyi Zheng, Rui Wen, Xi Chen, Yifan Yang, Yunyan Zhang, Ziheng Zhang, Ningyu Zhang, Bin Qin, Ming Xu, and Yefeng Zheng. 2021. Prgc: Potential relation and global correspondence based joint relational triple extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. 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 (Volume 1: Long Papers)*, pages 1227–1236.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 427–434.

## A Implementation Details

In our experiments, for all datasets, the batch size is 4 and the learning rate is set to 1e-5. The size of hidden state $h$ is 768. The max epoch is set to 150. The pre-trained model we used is [BERT-Base-Cased]. Following previous works (Fu et al., 2019; Wei et al., 2020; Zheng et al., 2021), the max length of input sentences to our model is set to 100 words. For our bias objective function, parameter $\beta$ and threshold $\lambda$ are two hyperparameters that are tuned together. For the NYT*, the threshold $\lambda$ is set to 0.8, and $\beta$ is set to 2. For WebNLG*, the threshold $\lambda$ is set to 0.92, and $\beta$ is set to 5. For our similarity gate, the threshold $\alpha$ we set is 0.5 for NYT* and 0.7 for WebNLG*. The number of layers of the fusion module is 2.

## B Analysis of Hyperparameter Setting

For bias objective function, we suggest parameter $\beta$ and threshold $\lambda$ together without any argument or evaluation supporting before, so an ablation study is done on WebNLG* with results reported in Table 8. It's not surprising to find that the performance of our model increases first and then decreases with the increasing value of $\beta$. F1 will peak when $\beta$ is between 2 and 6 on WebNLG* dataset. Note that the best result we get at $\beta$=5 on WebNLG* is heuristic, and a better F1 value might be obtained from another $\beta$ value between 2 and 6. Threshold $\lambda$ changes with $\beta$. Train with a fixed $\beta$, and test with different thresholds. The threshold $\lambda$ with the best results is almost the threshold $\lambda$ that best matches the $\beta$. The determination process of parameters parameter $\beta$ and threshold $\lambda$ of other datasets is similar to the above process on WebNLG*.

| $\beta$ | $\lambda$ | Prec. | Rec. | F1 |
|---|---|---|---|---|
| 1 | 0.5 | 94.2 | 93.4 | 93.8 |
| 2 | 0.74 | 94.5 | 93.4 | 94.0 |
| 5 | 0.92 | 95.0 | 93.6 | **94.3** |
| 6 | 0.93 | 94.8 | 93.7 | 94.2 |
| 10 | 0.98 | 94.5 | 93.9 | 94.2 |

Table 8: Some combinations of $\beta$ and $\lambda$ on WebNLG*.

## C Supplemental Experiments

We conduct a set of supplemental experiments to show the generalization capability in more general cases on two widely used datasets, namely, NYT10-HRL and NYT11-HRL. The results are reported in Table 9.

NYT corpus has two versions: (1) the original version of which both the training set and test set are produced via distant supervision by Riedel et al. (2010) and (2) a smaller version with fewer relation types, where the training set is produced by distant supervision while the test set is manually annotated by Hoffmann et al. (2011). We denote the original one and the smaller one as NYT10 and NYT11. These two versions have been selectively adopted and preprocessed in many different ways among various previous works, which may be confusing sometimes and lead to incomparable results if not specifying the version. To fairly compare these models, HRL (Takanobu et al., 2019) adopted a unified preprocessing for both NYT10 and NYT11, and provided a comprehensive comparison with previous works using the same datasets. Here we denote the preprocessed two versions as NYT10-HRL and NYT11-HRL.

| | Model | Prec. | Rec. | F1 |
|---|---|---|---|---|
| NYT10-HRL | NovelTagging(PM) | 59.3 | 38.1 | 46.4 |
| | CopyR (PM) | 56.9 | 45.2 | 50.4 |
| | CasRel(PM) | 77.7 | 68.8 | 73.0 |
| | Ours(PM) | **81.1** | **72.1** | **76.3** |
| | Ours(EM) | 80.4 | 71.6 | 75.7 |
| NYT11-HRL | NovelTagging(PM) | 96.9 | 48.9 | 47.9 |
| | CopyR(PM) | 34.7 | 53.4 | 42.1 |
| | CasRel(PM) | 50.1 | 58.4 | 53.9 |
| | Ours(PM) | **55.5** | **61.4** | **58.3** |
| | Ours(EM) | 55.0 | 60.8 | 57.8 |

Table 9: Relational triple extraction results on NYT10-HRL and NYT11-HRL.

For a fair comparison, we use the preprocessed datasets released by (Takanobu et al., 2019), where NYT10-HRL contains 70,339 sentences for training and 4,006 sentences for test and NYT11-HRL contains 62,648 sentences for training and 369 sentences for test. We also create a validation set by randomly sampling 0.5% data from the training set for each dataset as in (Takanobu et al., 2019; Wei et al., 2020). All the experimental results of the baseline models which use Partial Match (PM) are directly taken from Wei et al. (2020) unless specified. To keep in line with previous works, we use Partial Match for NYT10-HRL and NYT11-HRL. Since multiword entities are common in real-world scenarios, we also use Exact Match (EM) for the datasets.

When using Partial Match, there is a significant gap (from 73.0 to 76.3 in terms of F1-score on NYT10-HRL and from 53.9 to 58.0 in terms of F1-score on NYT11-HRL) between the performance of ours and CasRel. There is even a significant gap between ours using Exact Match and others using Partial Match.