A Novel Hierarchical Binary Tagging Framework for Relational Triple Extraction

Zhepei Wei[1], **Jianlin Su**[2], **Yue Wang**[3], **Yuan Tian**[1*], **Yi Chang**[1]
[1]School of Artificial Intelligence, Jilin University
[2]Shenzhen Zhuiyi Technology Co., Ltd.
[3]School of Information and Library Science, University of North Carolina at Chapel Hill
weizp19@mails.jlu.edu.cn, bojonesu@wezhuiyi.com, wangyue@email.unc.edu, {yuantian,yichang}@jlu.edu.cn
*Corresponding Author

Abstract

Extracting relational triples from unstructured text is crucial for large-scale knowledge graph construction. However, few existing works excel in solving the overlapping triple problem where multiple relational triples in the same sentence share the same entities. In this work, we introduce a fresh perspective to revisit the relational triple extraction task and propose a novel Hierarchical Binary Tagging (HBT) framework derived from a principled problem formulation. Instead of treating relations as discrete labels as in previous works, our new framework models relations as functions that map subjects to objects in a sentence, which naturally handles the overlapping problem. Experiments show that the proposed framework already outperforms state-of-the-art methods even when its encoder module uses a randomly initialized BERT encoder, showing the power of the new tagging framework. It enjoys further performance boost when employing a pretrained BERT encoder, outperforming the strongest baseline by 17.5 and 30.2 absolute gain in F1-score on two public datasets NYT and WebNLG, respectively. In-depth analysis on different scenarios of overlapping triples shows that the method delivers consistent performance gain across all these scenarios.

The source code is released online[1].

1 Introduction

The key ingredient of a knowledge graph is relational facts, most of which consist of two entities connected by a semantic relation. These facts are in the form of (subject, relation, object), or (s, r, o), referred to as relational triples. Extracting relational triples from natural language text is a crucial step towards constructing large-scale knowledge graphs.

[1]https://github.com/weizhepei/HBT

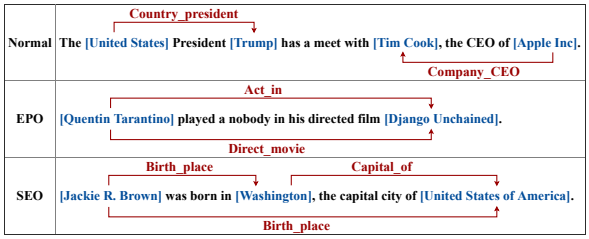


Figure 1: Examples of *Normal*, *EntityPairOverlap (EPO)* and *SingleEntityOverlap (SEO)* overlapping patterns.

Early works in relational triple extraction took a pipeline approach (Zelenko et al., 2003; Zhou et al., 2005; Chan and Roth, 2011). It first recognizes all entities in a sentence and then performs relation classification for each entity pair. Such an approach tends to suffer from the error propagation problem since errors in early stages cannot be corrected in later stages. To tackle this problem, subsequent works proposed joint learning of entities and relations, among them are feature-based models (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Ren et al., 2017) and, more recently, neural network-based models (Gupta et al., 2016; Zheng et al., 2017; Zeng et al., 2018; Fu et al., 2019). By replacing manually constructed features with learned representations, neural network-based models have achieved considerable success in the triple extraction task.

However, most existing approaches cannot properly handle scenarios in which a sentence contains multiple relational triples that overlap with each other. Figure 1 illustrates these scenarios, where triples share one or two entities in a sentence. This *overlapping triple problem* directly challenges conventional sequence tagging schemes that assume each token bears only one tag (Zheng et al., 2017). It also brings significant difficulty to relation classification approaches where an entity pair is as-

sumed to hold at most one relation (Katiyar and Cardie, 2017). Zeng et al. (2018) is among the first to consider the overlapping triple problem in relational triple extraction. They introduced the categories for different overlapping patterns as shown in Figure 1 and proposed a sequence-to-sequence (Seq2Seq) model with copy mechanism to extract triples. Based on the Seq2Seq model, they further investigate the impact of extraction order (Zeng et al., 2019) and gain considerable improvement with reinforcement learning. Fu et al. (2019) also studied the overlapping triple problem by modeling text as relational graphs with a graph convolutional networks (GCNs) based model.

Despite their success, previous works on extracting overlapping triples still leave much to be desired. Specifically, they all treat relations as discrete labels to be assigned to entity pairs. This formulation makes relation classification a hard machine learning problem. First, the class distribution is highly imbalanced. Among all pairs of extracted entities, most do not form valid relations, generating too many negative examples. Second, the classifier can be confused when the same entity participates in multiple valid relations (overlapping triples). Without enough training examples, the classifier can hardly tell which relation the entity participates in. As a result, the extracted triples are usually incomplete and inaccurate.

In this work, we start with a principled formulation of relational triple extraction right at the triple level. This gives rise to a general algorithmic framework that handles the overlapping triple problem by design. At the core of the framework is the fresh perspective that instead of treating relations as discrete labels on entity pairs, we can model relations as functions that map subjects to objects. More precisely, instead of learning relation classifiers $f(s, o) \rightarrow r$, we learn *relation-specific taggers* $f_r(s) \rightarrow o$, each of which recognizes the possible object(s) of a given subject under a specific relation; or returns no object, indicating that there is no triple with the given subject and relation. Under this framework, triple extraction is a two-step process: first we identify all possible subjects in a sentence; then for each subject, we apply relation-specific taggers to simultaneously identify all possible relations and the corresponding objects.

We implement the above idea in an end-to-end hierarchical binary tagging (HBT) framework. It consists of a BERT-based encoder module, a subject tagging module, and a relation-specific object tagging module. Empirical experiments show that the proposed framework outperforms state-of-the-art methods by a large margin even when the BERT encoder is *not* pretrained, showing the superiority of the new framework itself. The framework enjoys a further large performance gain after adopting a pretrained BERT encoder, showing the importance of rich prior knowledge in triple extraction task.

This work has the following main contributions:

1. We introduce a fresh perspective to revisit the relational triple extraction task with a principled problem formulation, which implies a general algorithmic framework that addresses the overlapping triple problem by design.

2. We instantiate the above framework as a novel hierarchical binary tagging model on top of a Transformer encoder. This allows the model to combine the power of the novel tagging framework with the prior knowledge in pretrained large-scale language models.

3. Extensive experiments on two public datasets show that the proposed framework overwhelmingly outperforms state-of-the-art methods, achieving 17.5 and 30.2 absolute gain in F1-score on the two datasets respectively. Detailed analyses show that our model gains consistent improvement in all scenarios.

2 Related Work

Extracting relational triples from unstructured natural language texts is a well-studied task in information extraction (IE). It is also an important step for the construction of large scale knowledge graph (KG) such as DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and Knowledge Vault (Dong et al., 2014).

Early works (Mintz et al., 2009; Gormley et al., 2015) address the task in a pipelined manner. They extract relational triples in two steps: 1) first run named entity recognition (NER) on the input sentence to identify all entities and 2) then run relation classification (RC) on pairs of extracted entities. The pipelined methods usually suffer from the error propagation problem since the second step is unavoidably affected by the errors introduced in the first step. To ease these issues, many joint models that aim to learn entities and relations jointly have been proposed. Traditional joint models (Yu

and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Ren et al., 2017) are feature-based, which heavily rely on feature engineering and require intensive manual efforts. To reduce manual work, recent studies have investigated neural network-based methods, which deliver state-of-the-art performance. However, most existing neural models (Miwa and Bansal, 2016; Katiyar and Cardie, 2017) achieve joint learning of entities and relations only through parameter sharing but not joint decoding. To obtain relational triples, they still have to pipeline the detected entity pairs to a relation classifier for identifying the relation of entities. The separated decoding setting leads to a separated training objective for entity and relation, which brings a drawback that the triple-level dependencies between predicted entities and relations cannot be fully exploited. Different from those works, Zheng et al. (2017) achieves joint decoding by introducing a novel unified tagging scheme and convert the task of relational triple extraction to an end-to-end sequence tagging problem without need of NER or RC. The proposed model can directly learn relational triples as a whole since the information of entities and relations is integrated into the unified tagging scheme.

Though joint models (with or without joint decoding) have been well studied, most previous works ignore the problem of overlapping relational triples. Zeng et al. (2018) introduced three patterns of overlapping triples and try to address the problem via a sequence-to-sequence model with copy mechanism. Recently, Fu et al. (2019) also studies the problem and propose a graph convolutional networks (GCNs) based method. Despite their initial success, both methods still treat the relations as discrete labels of entity pairs, making it quite hard for the model to learn overlapping triples.

Our framework is based on a training objective that is carefully designed to directly model the relational triples as a whole like (Zheng et al., 2017), i.e., to learn both entities and relations through joint decoding. Moreover, we model the relations as functions that map subjects to objects, which makes it crucially different from previous works.

3 Hierarchical Binary Tagging Framework

The goal of relational triple extraction is to identify all possible (subject, relation, object) triples in a sentence, where some triples may share the same entities as subjects or objects. Towards this goal, we directly model the triples and design a training objective right at the triple level. This is in contrast to previous approaches like (Fu et al., 2019) where the training objective is defined separately for entities and relations without explicitly modeling their integration at the triple level.

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 of the training set D:

$$\prod_{j=1}^{|D|} \left[\prod_{(s,r,o)\in T_j} p((s, r, o)|x_j) \right] \tag{1}$$

$$= \prod_{j=1}^{|D|} \left[\prod_{s\in T_j} p(s|x_j) \prod_{(r,o)\in T_j|s} p((r, o)|s, x_j) \right] \tag{2}$$

$$= \prod_{j=1}^{|D|} \left[\prod_{s\in T_j} p(s|x_j) \prod_{r\in T_j|s} p_r(o|s, x_j) \prod_{r\in R\setminus T_j|s} p_r(o_\varnothing|s, x_j) \right]. \tag{3}$$

Here we slightly abuse the notation T_j. $s \in T_j$ denotes a subject appearing in the triples in T_j. $T_j|s$ is the set of triples led by subject s in T_j. $(r, o) \in T_j|s$ is a (r, o) pair in the triples led by subject s in T_j. R is the set of all possible relations. $R\setminus T_j|s$ denotes all relations except those led by s in T_j. $o_\varnothing$ denotes a "null" object (explained below).

Eq. (2) applies the chain rule of probability. Eq. (3) exploits the crucial fact that for a given subject s, any relation relevant to s (those in $T_j|s$) would lead to corresponding objects in the sentence, and all other relations would necessarily have no object in the sentence, i.e. a "null" object.

This formulation provides several benefits. First, since the data likelihood starts at the triple level, optimizing this likelihood corresponds to directly optimizing the final evaluation criteria at the triple level. Second, by making no assumption on how multiple triples may share entities in a sentence, it handles the overlapping triple problem *by design*. Third, the decomposition in Eq. (3) inspires a novel tagging scheme for triple extraction: we learn a subject tagger $p(s|x_j)$ that recognizes subject entities in a sentence; and for each relation r, we learn an object tagger $p_r(o|s, x_j)$ that recognizes relation-specific objects for a given subject. In this way we can model each relation as a function that maps subjects to objects, as opposed to classifying relations for (*subject*, *object*) pairs.

Indeed, this novel tagging scheme allows us to extract multiple triples at once: we first run the

subject tagger to find all possible subjects in the sentence, and then for each subject found, apply relation-specific object taggers to find all relevant relations and the corresponding objects.

The key components in the above general framework, i.e., the subject tagger and relation-specific object taggers, can be instantiated in many ways. In this paper, we instantiate them as binary taggers on top of a deep bidirectional Transformer BERT (Devlin et al., 2019). We describe its detail below.

3.1 BERT Encoder

The encoder module extracts feature information $\mathbf{x}_j$ from sentence x_j, which will feed into subsequent tagging modules[2]. We employ a pretrained BERT model (Devlin et al., 2019) to encode the context information.

Here we briefly review BERT, a multi-layer bidirectional Transformer based language representation model. It is designed to learn deep representations by jointly conditioning on both left and right context of each word and it has recently been proven surprisingly effective in many downstream tasks. Specifically, it is composed of a stack of N identical Transformer blocks. We denote the Transformer block as $Trans(\mathbf{x})$, in which $\mathbf{x}$ represents the input vector. The detailed operations are as follows:

$$\mathbf{h}_0 = \mathbf{S}\mathbf{W}_s + \mathbf{W}_p \tag{4}$$

$$\mathbf{h}_\alpha = Trans(\mathbf{h}_{\alpha-1}), \alpha \in [1, N] \tag{5}$$

where $\mathbf{S}$ is the matrix of one-hot vectors of sub-words[3] indices in the input sentence, $\mathbf{W}_s$ is the sub-words embedding matrix, $\mathbf{W}_p$ is the positional embedding matrix where p represents the position index in the input sequence, $\mathbf{h}_\alpha$ is the hidden state vector, i.e., the context representation of input sentence at α-th layer and N is the number of Transformer blocks. Note that in our work the input is a single text sentence instead of sentence pair, hence the segmentation embedding as described in original BERT paper was not taken into account in Eq. (4). For a more comprehensive description of the Transformer structure, we refer readers to (Vaswani et al., 2017).

[2]This paper uses boldface letters to denote vectors and matrices.

[3]We use WordPiece embeddings (Wu et al., 2016) to represent words in vector space as in BERT (Devlin et al., 2019). Each word in the input sentence will be tokenized to fine-grained tokens, *i.e.*, sub-words.

3.2 Hierarchical Decoder

Now we describe our instantiation of the novel hierarchical binary tagging scheme inspired by the previous formulation. The basic idea is to extract triples in two steps. First, we detect subjects from the input sentence. Then for each candidate subject, we check all possible relations to see if a relation can associate objects in the sentence with that subject. Corresponding to the two steps, the hierarchical decoder consists of two modules as illustrated in Figure 2: a subject tagger; and a set of relation-specific object taggers.

Subject Tagger The low level tagging module is designed to recognize all possible subjects in the input sentence by directly decoding the encoded vector $\mathbf{h}_N$ produced by the N-layer BERT encoder. More precisely, it adopts two identical binary classifiers to detect the start and end position of subjects respectively by assigning each token a binary tag (0/1) that indicates whether the current token corresponds to a start or end position of a subject. The detailed operations of the subject tagger on each token are as follows:

$$p_i^{start\_s} = \sigma(\mathbf{W}_{start}\mathbf{x}_i + \mathbf{b}_{start}) \tag{6}$$

$$p_i^{end\_s} = \sigma(\mathbf{W}_{end}\mathbf{x}_i + \mathbf{b}_{end}) \tag{7}$$

where $p_i^{start\_s}$ and $p_i^{end\_s}$ represent the probability of identifying the i-th token in the input sequence as the start and end position of a subject, respectively. The corresponding token will be assigned with a tag 1 if the probability exceeds a certain threshold or with a tag 0 otherwise. $\mathbf{x}_i$ is the encoded representation of the i-th token in the input sequence, *i.e.*, $\mathbf{x}_i = \mathbf{h}_N[i]$, where $\mathbf{W}_{(\cdot)}$ represents the trainable weight, and $\mathbf{b}_{(\cdot)}$ is the bias and σ is the sigmoid activation function.

The subject tagger optimizes the following likelihood function to identify the span of subject s given a sentence representation $\mathbf{x}$:

$$p_\theta(s|\mathbf{x}) = \prod_{t \in \{start\_s, end\_s\}} \prod_{i=1}^{L} \left(p_i^t\right)^{\mathbf{I}\{y_i^t=1\}} \left(1 - p_i^t\right)^{\mathbf{I}\{y_i^t=0\}}. \tag{8}$$

where L is the length of the sentence. $\mathbf{I}\{z\} = 1$ if z is true and 0 otherwise. $y_i^{start\_s}$ is the binary tag of subject start position for the i-th token in $\mathbf{x}$, and $y_i^{end\_s}$ indicates the subject end position. The parameters $\theta = \{\mathbf{W}_{start}, \mathbf{b}_{start}, \mathbf{W}_{end}, \mathbf{b}_{end}\}$.

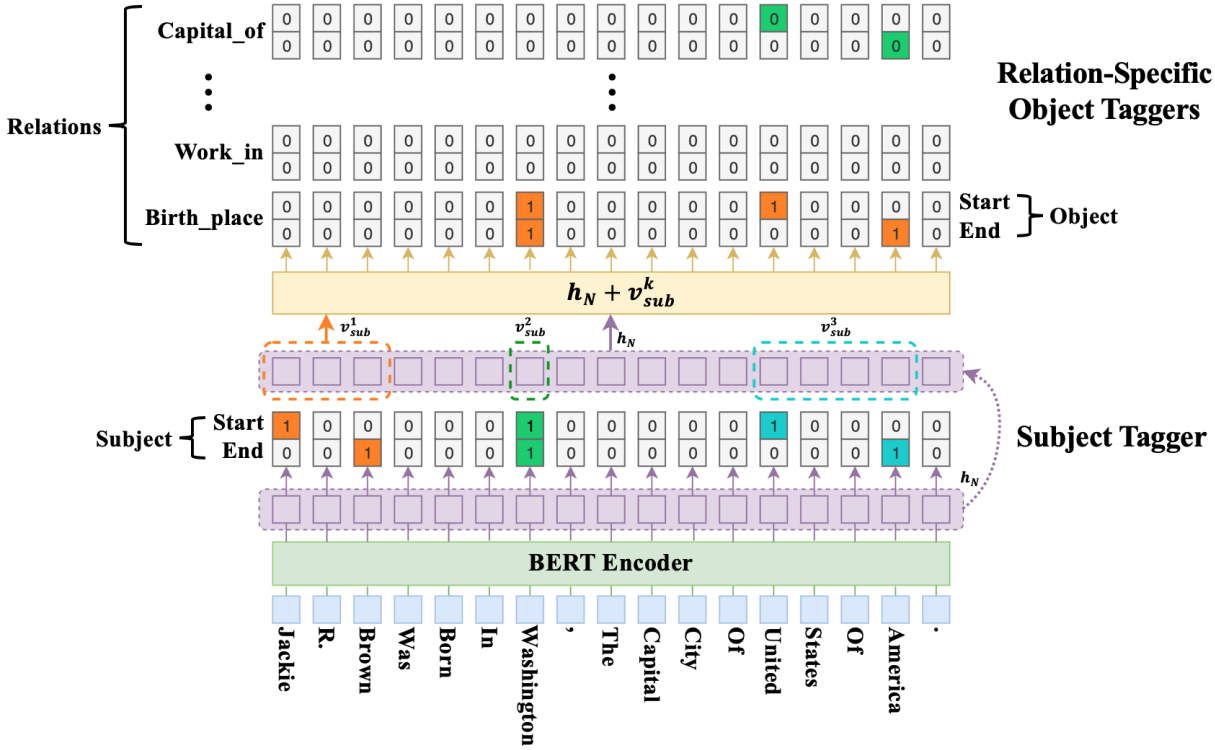


Figure 2: An overview of the proposed hierarchical binary tagging (HBT) framework structure. In this example, there are three candidate subjects detected at the low level, while the presented 0/1 tags at high level are specific to the first subject *Jackie R. Brown*, i.e., a snapshot of the iteration state when $k = 1$ is shown as above. For the subsequent iterations ($k = 2, 3$), the results at high level will change, reflecting different triples detected. For instance, when $k = 2$, the high-level orange (green) blocks will change to 0 (1), respectively, reflecting the relational triple *(Washington, Capital_of, United States Of America)* led by the second candidate subject *Washington*.

For multiple subjects detection, we adopt the nearest start-end pair match principle to decide the span of any subject based on the results of the start and end position taggers. For example, as shown in Figure 2, the nearest end token to the first start token "Jackie" is "Brown", hence the detected result of the first subject span will be "Jackie R. Brown". Noticeably, to match an end token for a given start token, we don't consider tokens whose position is prior to the position of the given token. Such match strategy is able to maintain the integrity of any entity span if the start and end positions are both correctly detected due to the natural continuity of any entity span in a given sentence.

Relation-specific Object Taggers The high level tagging module simultaneously identifies the objects as well the involved relations with respect to the subjects obtained at lower level. As Figure 2 shows, it consists of a set of relation-specific object taggers with the same structure as subject tagger in low level module for all possible relations. All object taggers will identify the corresponding object(s) for each detected subject at the same time. Different from subject tagger directly decoding the encoded vector $\mathbf{h}_N$, the relation-specific object tagger takes the subject features into account as well. The detailed operations of the relation-specific object tagger on each token are as follows:

$$p_i^{start\_o} = \sigma(\mathbf{W}^r_{start}(\mathbf{x}_i + \mathbf{v}^k_{sub}) + \mathbf{b}^r_{start}) \quad (9)$$

$$p_i^{end\_o} = \sigma(\mathbf{W}^r_{end}(\mathbf{x}_i + \mathbf{v}^k_{sub}) + \mathbf{b}^r_{end}) \quad (10)$$

where $p_i^{start\_o}$ and $p_i^{end\_o}$ represent the probability of identifying the i-th token in the input sequence as the start and end position of a object respectively, and $\mathbf{v}^k_{sub}$ represents the encoded representation vector of the k-th subject detected in low level module.

For each subject, we iteratively apply the same decoding process on it. Note that the subject is usually composed of multiple tokens, to make the additions of $\mathbf{x}_i$ and $\mathbf{v}^k_{sub}$ in Eq. (9) and Eq. (10) possible, we need to keep the dimension of two vectors consistent. To do so, we take the averaged vector representation of all the tokens that the k-th subject contains as $\mathbf{v}^k_{sub}$.

The object tagger for relation r optimizes the following likelihood function to identify the span

of object o given a sentence representation $\mathbf{x}$ and a subject s:

$$p_{\phi_r}(o|s,\mathbf{x}) = \prod_{t\in\{start\_o,end\_o\}} \prod_{i=1}^{L} \left(p_i^t\right)^{\mathbf{I}\{y_i^t=1\}} \left(1-p_i^t\right)^{\mathbf{I}\{y_i^t=0\}} . \quad (11)$$

where $y_i^{start\_o}$ is the binary tag of object start position for the i-th token in $\mathbf{x}$, and $y_i^{end\_o}$ is the tag of object end position for the i-th token. For a "null" object $o_\varnothing$, the tags $y_i^{start\_o_\varnothing} = y_i^{end\_o_\varnothing} = 0$ for all i. The parameters $\phi_r = \{\mathbf{W}_{start}^r, \mathbf{b}_{start}^r, \mathbf{W}_{end}^r, \mathbf{b}_{end}^r\}$.

Note that in the high level tagging module, the relation is also decided by the output of object taggers. For example, the relation "Work_in" does not hold between the detected subject "Jackie R. Brown" and the candidate object "Washington". Therefore, the object tagger for relation "Work_in" will not identify the span of "Washington", i.e., the output of both start and end position are all zeros as shown in Figure 2. In contrast, the relation "Birth_place" holds between "Jackie R. Brown" and "Washington", so the corresponding object tagger outputs the span of the candidate object "Washington". In this setting, the high level module is capable of simultaneously identifying the relations and objects with regard to the subjects detected in low level module.

3.3 Data Log-likelihood Objective

Taking log of Eq. (3), the objective $J(\Theta)$ is:

$$\sum_{j=1}^{|D|} \left[\sum_{s\in T_j} \log p_\theta(s|\mathbf{x}_j) + \sum_{r\in T_j|s} \log p_{\phi_r}(o|s,\mathbf{x}_j) + \sum_{r\in R\setminus T_j|s} \log p_{\phi_r}(o_\varnothing|s,\mathbf{x}_j) \right] . \quad (12)$$

where parameters $\Theta = \{\theta, \{\phi_r\}_{r\in R}\}$. $p_\theta(s|\mathbf{x})$ is defined in Eq. (8) and $p_{\phi_r}(o|s,\mathbf{x})$ is defined in Eq. (11). We train the model by maximizing $J(\Theta)$ through Adam stochastic gradient descent over shuffled mini-batches (Kingma and Ba, 2014).

4 Experiments

4.1 Experimental Setting

Datasets and Evaluation Metrics We evaluate the proposed HBT framework on two public datasets NYT (Riedel et al., 2010) and WebNLG (Gardent et al., 2017). NYT dataset was originally produced by distant supervision method. It consists of 1.18M sentences with 24 predefined relation types. WebNLG dataset was originally created for Natural Language Generation (NLG) tasks and adapted by (Zeng et al., 2018) for relational triple extraction task. It contains 246 predefined relation types. The sentences in both datasets commonly contain multiple relational triples, thus NYT and WebNLG datasets suit very well to be the testbed for evaluating model on extracting overlapping relational triples[4]. We use the datasets released by (Zeng et al., 2018), in which NYT contains 56195 sentences for training, 5000 sentences for validation, and 5000 sentences for test, and WebNLG contains 5019 sentences for training, 500 sentences for validation and 703 sentences for test. According to different overlapping patterns of relational triples, we split the sentences into three categories, namely, *Normal*, *EntityPairOverlap (EPO)* and *SingleEntityOverlap (SEO)* for detailed experiments on different types of overlapping relational triples. The statistics of the two datasets are described in Table 1.

Category	NYT		WebNLG	
	Train	Test	Train	Test
Normal	37013	3266	1596	246
EPO	9782	978	227	26
SEO	14735	1297	3406	457
ALL	56195	5000	5019	703

Table 1: Statistics of datasets. Note that a sentence can belong to both *EPO* class and *SEO* class.

Following previous work (Fu et al., 2019), an extracted relational triple *(subject, relation, object)* is regarded as correct only if the relation and the heads of both subject and object are all correct. For fair comparison, we report the standard micro Precision (Prec.), Recall (Rec.) and F1-score as in line with baselines.

Implementation Details The hyper-parameters are determined on the validation set. More implementation details are described in Appendix A.

[4]Datasets such as ACE, Wiki-KBP have few overlapping triples in the sentences hence are not suitable for evaluating the performance of overlapping triple extraction. Nonetheless, to validate the generality of the proposed framework, we also conduct supplemental experiments on these datasets along with the comparison of 13 recent strong baselines. The results of the comprehensive comparison, which show consistent superiority of our model over most compared methods, can be found in Appendix C.

Method	NYT			WebNLG		
	Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
NovelTagging (Zheng et al., 2017)	62.4	31.7	42.0	52.5	19.3	28.3
CopyR$_{OneDecoder}$ (Zeng et al., 2018)	59.4	53.1	56.0	32.2	28.9	30.5
CopyR$_{MultiDecoder}$ (Zeng et al., 2018)	61.0	56.6	58.7	37.7	36.4	37.1
GraphRel$_{1p}$ (Fu et al., 2019)	62.9	57.3	60.0	42.3	39.2	40.7
GraphRel$_{2p}$ (Fu et al., 2019)	63.9	60.0	61.9	44.7	41.1	42.9
CopyR$_{RL}$ (Zeng et al., 2019)	77.9	67.2	72.1	63.3	59.9	61.6
CopyR$^{*}_{RL}$	72.8	69.4	71.1	60.9	61.1	61.0
HBT$_{random}$	81.5	75.7	78.5	84.7	79.5	82.0
HBT$_{LSTM}$	84.2	83.0	83.6	86.9	80.6	83.7
HBT	**89.7**	**89.5**	**89.6**	**93.4**	**90.1**	**91.8**

Table 2: Results of different methods on NYT and WebNLG datasets. Our re-implementation is marked by *.

4.2 Experimental Result

Compared Methods We compare our model with several strong state-of-the-art models, namely, **NovelTagging** (Zheng et al., 2017), **CopyR** (Zeng et al., 2018), **GraphRel** (Fu et al., 2019) and **CopyR**$_{RL}$ (Zeng et al., 2019). The reported results for the above baselines are directly copied from the original published literature.

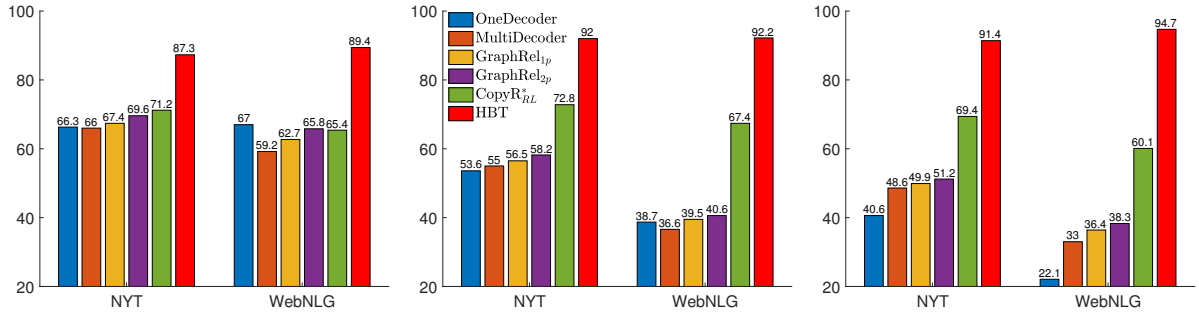
Note that we instantiate the HBT framework on top of a pre-trained BERT model to combine the power of the proposed novel tagging scheme and the pre-learned prior knowledge for better performance. To evaluate the impact of introducing the Transformer-based BERT model in our method, we conduct a set of ablation tests. **HBT**$_{random}$ is the framework where all parameters of BERT are randomly initialized; **HBT**$_{LSTM}$ is the framework instantiated on a LSTM-based structure as in (Zheng et al., 2017) with pretrained Glove embedding (Pennington et al., 2014); **HBT** is the full-fledged framework using pre-trained BERT weights.

Main Results Table 2 shows the results of different baselines for relational triple extraction on two datasets. The HBT model overwhelmingly outperforms all the baselines in terms of all three evaluation metrics and achieves encouraging 17.5% and 30.2% improvements in F1-score over the best state-of-the-art method (Zeng et al., 2019) on NYT and WebNLG datasets respectively. Even without taking advantage of the pre-trained BERT, HBT$_{random}$ and HBT$_{LSTM}$ are still competitive to existing state-of-the-art models. This validates the utility of the proposed hierarchical decoder that adopts a novel binary tagging scheme. The performance improvements from HBT$_{random}$ to HBT highlight the importance of the prior knowledge in a pre-trained language model.

We can also observe from the table that there is a significant gap between the performance on NYT and WebNLG datasets for existing models and we believe this gap is due to their drawbacks in dealing with overlapping triples. More precisely, as presented in Table 1, we can find that NYT dataset is mainly comprised of *Normal* class sentences while the majority of sentences in WebNLG dataset belong to *EPO* and *SEO* classes. Such inconsistent data distribution of two datasets leads to a comparatively better performance on NYT and a worse performance on WebNLG for all the baselines, exposing their drawbacks in extracting overlapping relational triples. In contrast, the HBT model and its variants (i.e., HBT$_{random}$ and HBT$_{LSTM}$) all achieve a stable and competitive performance on both NYT and WebNLG datasets, demonstrating the effectiveness of the proposed framework in solving the overlapping problem.

Detailed Results on Different Types of Sentences To further study the capability of the proposed HBT framework in extracting overlapping relational triples, we conduct two extended experiments on different types of sentences and compare the performance with previous works.

The detailed results on three different overlapping patterns are presented in Figure 3. It can be seen that the performance of most baselines on *Normal*, *EPO* and *SEO* presents a decreasing trend, reflecting the increasing difficulty of extracting relational triples from sentences with different overlapping patterns. That is, among the three overlapping patterns, *Normal* class is the easiest pattern while *EPO* and *SPO* classes are the relatively harder ones for baseline models to extract. In contrast, the proposed HBT method attains consistently strong performance over all three overlapping patterns, es-



(a) F1 of Normal Class (b) F1 of EntityPairOverlap Class (c) F1 of SingleEntityOverlap Class

Figure 3: F1-score of extracting relational triples from sentences with different overlapping pattern.

Method	NYT					WebNLG				
	N=1	*N=2*	*N=3*	*N=4*	*N≥5*	*N=1*	*N=2*	*N=3*	*N=4*	*N≥5*
CopyR$_{OneDecoder}$	66.6	52.6	49.7	48.7	20.3	65.2	33.0	22.2	14.2	13.2
CopyR$_{MultiDecoder}$	67.1	58.6	52.0	53.6	30.0	59.2	42.5	31.7	24.2	30.0
GraphRel$_{1p}$	69.1	59.5	54.4	53.9	37.5	63.8	46.3	34.7	30.8	29.4
GraphRel$_{2p}$	71.0	61.5	57.4	55.1	41.1	66.0	48.3	37.0	32.1	32.1
CopyR$^{*}_{RL}$	71.7	72.6	72.5	77.9	45.9	63.4	62.2	64.4	57.2	55.7
HBT	**88.2**	**90.3**	**91.9**	**94.2**	**83.7 (+37.8)**	**89.3**	**90.8**	**94.2**	**92.4**	**90.9 (+35.2)**

Table 3: F1-score of extracting relational triples from sentences with different number (denoted as N) of triples.

pecially for those hard patterns. We also validate the HBT's capability in extracting relational triples from sentences with different number of triples. We split the sentences into five classes and Table 3 shows the results. Again, our HBT model achieves excellent performance over all five classes. Though it's not surprising to find that the performance of most baselines decrease with the increasing number of relational triples that a sentence contains, some patterns still can be observed from the performance changes of different models. Compared to previous works that devote to solving the overlapping problem in relational triple extraction, our model suffers the least from the increasing complexity of the input sentence. Though our HBT model gain considerable improvements on all five classes compared to the best state-of-the-art method CopyR$_{RL}$ (Zeng et al., 2019), the greatest improvement of F1-score on the two datasets (i.e., 37.8 on NYT and 35.2 on WebNLG) both come from the most difficult class (N≥5), indicating that our model is more suitable for complicated scenarios than the baselines.

Both of these experiments validate the superiority of the proposed hierarchical binary tagging framework in extracting multiple (possibly overlapping) relational triples from complicated sentences compared to existing methods. Previous works have to explicitly predict all possible relation types contained in a given sentence, which is quite a challenging task, and thus many relations are missing in their extracted results. In contrast, our HBT model side-steps the prediction of relation types and tends to extract as many relational triples as possible from a given sentence. We attribute this to the relation-specific object tagger setting in high level tagging module of the hierarchical decoder that considers all the relation types simultaneously.

5 Conclusion

In this paper, we introduce a novel hierarchical binary tagging (HBT) framework derived from a principled problem formulation for relational triple extraction. Instead of modeling relations as discrete labels of entity pairs, we model the relations as functions that map subjects to objects, which provides a fresh perspective to revisit the relational triple extraction task. As a consequent, our model can simultaneously extract multiple relational triples from sentences, without suffering from the overlapping problem. We conduct extensive experiments on two widely used datasets to validate the effectiveness of the proposed HBT framework. Experimental results show that our model overwhelmingly outperforms state-of-the-art baselines over different scenarios, especially on the extraction of overlapping relational triples.

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 *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, pages 722–735.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2830–2836.

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-Volume 1*, pages 551–560. Association for Computational Linguistics.

Dai Dai, Xinyan Xiao, Yajuan Lyu, Shan Dou, Qiaoqiao She, and Haifeng Wang. 2019. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6300–6308.

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.

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. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1774–1784.

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-Volume 1*, 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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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)*, volume 1, pages 402–412.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

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: Volume 2-Volume 2*, 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)*, volume 1, 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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *AAAI*.

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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1399–1407. Association for Computational Linguistics.

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

Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao. 2019. Learning the extraction order of multiple relational facts in a sentence with reinforcement learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 367–377.

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)*, volume 1, pages 506–514.

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)*, volume 1, 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 (ACL05)*, pages 427–434.

A Implementation Details

We adopt mini-batch mechanism to train our model with batch size as 6; the learning rate is set to $1e^{-5}$; the hyper-parameters are determined on the validation set. We also adopt early stopping mechanism to prevent the model from over-fitting. Specifically, we stop the training process when the performance on validation set does not gain any improvement for at least 7 consecutive epochs. The number of stacked bidirectional Transformer blocks N is 12 and the size of hidden state $\mathbf{h}_N$ is 768. The pre-trained BERT model we used is [BERT-Base, Cased][5],which contains 110M parameters.

For fair comparison, the max length of input sentence to our model is set to 100 words as previous works (Zeng et al., 2018; Fu et al., 2019) suggest. We did not tune the threshold for both start and end position taggers to predict tag 1, but heuristically set the threshold to 0.5 as default. The performance might be better after carefully tuning the threshold, however it is beyond the research scope of this paper.

B Error Analysis

To explore the factors that affect the extracted relational triples of our HBT model, 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

[5] Available at: https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip

Element	NYT			WebNLG		
	Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
E1	94.6	92.4	93.5	98.7	92.8	95.7
E2	94.1	93.0	93.5	97.7	93.0	95.3
R	96.0	93.8	94.9	96.6	91.5	94.0
(E1, R)	93.6	90.9	92.2	94.8	90.3	92.5
(R, E2)	93.1	91.3	92.2	95.4	91.1	93.2
(E1, E2)	89.2	90.1	89.7	95.3	91.7	93.5
(E1, R, E2)	89.7	89.5	89.6	93.4	90.1	91.8

Table 4: Results on relational triple elements.

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 is *E2* and *R*.

Table 4 shows the results on different relational triple elements. For NYT, the performance on *E1* and *E2* are consistent with that on *(E1, R)* and *(R, E2)*, demonstrating the effectiveness of our proposed framework in identifying both subject and object entity mentions. We also find that there is only a trivial gap between the F1-score on *(E1, E2)* and *(E1, R, E2)*, but an obvious gap between *(E1, R, E2)* and *(E1, R)/(R, E2)*. It reveals that most relations for the entity pairs in extracted triples are correctly identified while some extracted entities fail to form a valid relational triple. In other words, it implies that identifying relations is somehow easier than identifying entities for our model.

In contrast to NYT, for WebNLG, the performance gap between *(E1, E2)* and *(E1, R, E2)* is comparatively larger than that between *(E1, R, E2)* and *(E1, R)/(R, E2)*. It shows that misidentifying the relations will bring more performance degradation than misidentifying the entities. Such observation also indicates that it's more challenging for the proposed HBT model to identify the relations than entities in WebNLG, as opposed to what we observed in NYT. We attribute such difference to the different number of relations contained in the two datasets (i.e, 24 in NYT and 246 in WebNLG), which makes the identification of relation much harder in WebNLG.

C Supplemental Experiments

In addition to validating the effectiveness of the proposed HBT framework in handling the *overlapping triple problem*, we also conduct a set of supplemental experiments to show the generalization capability in more general cases on four widely used datasets, namely, ACE04, NYT10-HRL, NYT11-HRL and Wiki-KBP. Unlike the datasets we adopt in the main experiment, most test sentences in these datasets belong to the *Normal* class where no triples overlap with each other. Table 5 shows the result of a comprehensive comparison with recent state-of-the-art methods.

Noticeably, there are two different evaluation metrics selectively adopted among previous works: (1) The widely used one is ***Partial Match*** as we described in Section 4.1, i.e., an extracted relational triple *(subject, relation, object)* is regarded as correct only if the relation and the heads of both subject and object are all correct (Li and Ji, 2014; Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Zheng et al., 2017; Zeng et al., 2018; Takanobu et al., 2019; Li et al., 2019; Fu et al., 2019); (2) The stricter but less popular one is ***Exact Match*** adopted by Dai et al. (2019), where an extracted relational triple *(subject, relation, object)* is regarded as correct only if the relation and the heads and tails of both subject and object are all correct.

Since some works like (Zeng et al., 2018) can't handle multi-word entities and can only be evaluated under the *Partial Match* metric and some works like (Dai et al., 2019) are not open-source, it's hard to use a unified metric to compare our model with existing models. To properly compare our model with various baselines, we adopt the *Partial Match* metric for ACE04, NYT10-HRL and NYT11-HRL, and adopt the *Exact Match* metric for Wiki-KBP.

ACE04 We follow the same 5-fold cross-validation setting as adopted in previous works (Li and Ji, 2014; Miwa and Bansal, 2016; Li et al., 2019) and use the code[6] released by (Miwa and Bansal, 2016) to preprocess the raw XML-style data for fair comparison. Eventually, it has 2,171 valid sentences in total and each sentence contains at least one relational triple.

NYT10-HRL & NYT11-HRL NYT corpus have 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). Here we denote the original one and the smaller one as NYT10 and NYT11, respectively. These two versions have been selec-

[6]https://github.com/tticoin/LSTM-ER

Method	*Partial Match*									*Exact Match*		
	ACE04			NYT10-HRL			NYT11-HRL			Wiki-KBP		
	Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
MultiR (Hoffmann et al., 2011)	–	–	–	–	–	–	32.8	30.6	31.7	30.1	53.0	38.0
DS-Joint (Li and Ji, 2014)	**60.8**	36.1	45.3	–	–	–	–	–	–	–	–	–
FCM (Gormley et al., 2015)	–	–	–	–	–	–	43.2	29.4	35.0	–	–	–
SPTree (Miwa and Bansal, 2016)	48.7	48.1	48.4	49.2	55.7	52.2	52.2	54.1	53.1	–	–	–
CoType (Ren et al., 2017)	–	–	–	–	–	–	48.6	38.6	43.0	31.1	**53.7**	38.8
Katiyar and Cardie (2017)	46.4	45.3	45.7	–	–	–	–	–	–	–	–	–
NovelTagging (Zheng et al., 2017)	–	–	–	59.3	38.1	46.4	46.9	48.9	47.9	**53.6**	30.3	38.7
ReHession (Liu et al., 2017)	–	–	–	–	–	–	–	–	–	36.7	49.3	42.1
Bekoulis et al. (2018)	–	–	47.5	–	–	–	–	–	–	–	–	–
CopyR (Zeng et al., 2018)	–	–	–	56.9	45.2	50.4	34.7	53.4	42.1	–	–	–
HRL (Takanobu et al., 2019)	–	–	–	71.4	58.6	64.4	**53.8**	53.8	53.8	–	–	–
PA-LSTM-CRF (Dai et al., 2019)	–	–	–	–	–	–	–	–	–	51.1	39.3	44.4
MultiQA (Li et al., 2019)	50.1	**48.7**	49.4	–	–	–	–	–	–	–	–	–
HBT	57.2	47.6	**52.0**	**77.7**	**68.8**	**73.0**	50.1	**58.4**	**53.9**	49.8	42.7	**45.9**

Table 5: Triple extraction results of different methods under *Partial Match* and *Exact Match* metrics.

Category	ACE04	NYT10-HRL		NYT11-HRL		Wiki-KBP	
	ALL	Train	Test	Train	Test	Train	Test
Normal	1604	59396	2963	53395	368	57020	265
EPO	8	5376	715	2100	0	3217	4
SEO	561	8772	742	7365	1	21238	20
ALL	2171	70339	4006	62648	369	79934	289

Table 6: Statistics of datasets. Note that a sentence can belong to both *EPO* class and *SEO* class.

tively 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.

For 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).

Wiki-KBP We use the same version as Dai et al. (2019) adopted, where the training set is from (Liu et al., 2017) and the test set is from (Ren et al., 2017). It has 79,934 sentences for training and 289 sentences for test. We also create a validation set by randomly sampling 10% data from the test set as Dai et al. (2019) suggested.

Dataset Study As stated beforehand, these datasets are not suitable for testing the overlapping problem. To further explain this argument, we analyze the datasets in detail and the statistics are shown in Table 6. We find that the test data in these datasets suffer little from the so-called *overlapping triple problem* since the sentences contain few overlapping triples. Even worse, we also find that the annotated triples are not always the ground truth, which may affect the evaluation of our model. Take a real instance from the test set of NYT11-HRL for example, for the sentence *"Mr. Gates , who arrived in Paris on Tuesday , was the first American defense secretary to visit France in nearly a decade ."*, the annotated triple set only contains one relational triple: {*(Paris, /location/administrative_division/country, France)*}. However the output of our model has three triples: {*(Paris, /location/administrative_division/country, France), (France, /location/country/administrative_divisions, Paris), (France, /location/location/contains, Paris)*}. The last two triples should have been annotated in the sentence but were omitted, which will significantly affect the values of both precision and recall when quantifying the performance of our model.

This observation demonstrates that our HBT model could extract more relational triples than the manually annotated ones in some cases due to the imperfect annotation. For this reason, the performance on such dataset like NYT11-HRL can only partially reflect the potential of the proposed model and probably underestimate the real value. Nonetheless, the HBT model still achieves a competitive performance, showing the effectiveness of the proposed novel hierarchical binary tagging framework in relational triple extraction.

We also note that there is a significant gap (from 53.9 to 89.6 in terms of F1-score) between the performance on NYT11-HRL that preprocessed by Takanobu et al. (2019) and on NYT11-CopyR[7] that preprocessed by Zeng et al. (2018). Though both of the above two versions are adapted from the original NYT11 dataset (Hoffmann et al., 2011), there are two key differences in the NYT11-CopyR version as Dai et al. (2019) pointed out. First, instead of using the manually annotated test set, Zeng et al. (2018) randomly select 5000 sentences from the training data as the test data. The reason is that the manually annotated data contains few overlapping triples and thus not suitable for testing the overlapping triple problem; Second, Zeng et al. (2018) only annotated the **last word** of an entity in both training and test sets because their model can not handle multi-word entities. Hence, any entity in their dataset is taken as a single-word entity, leading to that the *Partial Match* and *Exact Match* evaluation metrics make no difference. Moreover, such setting makes it much easier for our HBT model to detect the span of an entity since the start and end positions are actually the same. We attribute the significant gap to these different settings between the above two versions. Noticeably, multi-word entities are common in real-world scenarios, so evaluating on a more proper dataset like NYT10-HRL (which aslo contains overlapping triples in test set) may better reveal the model's real value in relational triple extraction than on the ad-hoc one.

[7]We denote the version preprocessed by Zeng et al. (2018) as NYT11-CopyR for disambiguation, which is also referred to as "NYT" in the main experiment.