Frontiers of Computer Science CN 10-1014/TP
https://doi.org/10.1007/s11704-020-9420-6 ISSN 2095-2228
RESEARCH ARTICLE

Nested Relation Extraction with Iterative Neural Network

Yixuan CAO[1,2], Dian CHEN[1,2], Zhengqi XU[1,2], Hongwei LI[1,2], Ping LUO[1,2]

1 Key Lab of Intelligent Information Processing of Chinese Academy of Sciences(CAS), Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
2 University of Chinese Academy of Sciences, Beijing 100049, China

Front. Comput. Sci., **Just Accepted Manuscript** • 10.1007/s11704-020-9420-6
http://journal.hep.com.cn on March 27, 2020

Front.Comput.Sci.
https://doi.org/10.1007/s11704-020-9420-6

RESEARCH ARTICLE

Nested Relation Extraction with Iterative Neural Network

Yixuan CAO[1,2], Dian CHEN[1,2] , Zhengqi XU[1,2], Hongwei LI[1,2] , Ping LUO[1,2] (✉)

1 Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
2 University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

Most existing researches on relation extraction focus on binary flat relations like BornIn relation between a Person and a Location. But a large portion of objective facts described in natural language are complex, especially in professional documents in fields such as finance and biomedicine that require precise expressions. For example, "the GDP of the United States in 2018 grew 2.9% compared with 2017" describes a growth rate relation between two other relations about the economic index, which is beyond the expressive power of binary flat relations. Thus, we propose the Nested Relation Extraction problem and formulate it as a Directed Acyclic Graph (DAG) structure extraction problem. Then, we propose a solution using the Iterative Neural Network which extracts relations layer by layer. The proposed solution achieves 78.98 and 97.89 F1 scores on two nested relation extraction tasks, namely semantic cause-and-effect relation extraction and formula extraction. Furthermore, we observe that nested relations are usually expressed in long sentences where entities are mentioned repetitively, which makes the annotation difficult and error-prone. Hence, we extend our model to incorporate a mention-insensitive mode that only requires annotations of relations on entity concepts (instead of exact mentions) while preserving most of its performance. Our mention-insensitive model performs better than the mention sensitive model when the random level in mention selection is higher than 0.3.

Keywords Nested Relation Extraction, Mention Insensitive Relation, Iterative Neural Network

Received month dd, yyyy; accepted month dd, yyyy

E-mail: luop@ict.ac.cn

1 Introduction

Relation Extraction (RE) is the task of determining whether there are relations among some of the entities, and what the relation types are, given a sentence and entities (mentions or concepts) in this sentence.

Flat RE tasks extract relation among entities. Most flat RE tasks extract binary relations, i.e. relation between two entities, like BornIn relation between Trump and New York City from the sentence "Trump was born in New York City in 1946."). Their results are the foundation and data source for many downstream tasks such as knowledge base construction and question answering. However, facts described in natural language are beyond binary flat relations due to the complex nature of language (as exemplified in the following paragraphs). We summarize two aspects of the complexity of relation: high-arity [1] and nested. *High-arity* extends the concept of relation to be among multiple entities. *Nested* extends the concept of relation to be upon other relations. Complex relations are expressed in natural language frequently, especially in economics, finance, biomedicine and other fields where people need to express in a precise way. We show one example for each of the two aspects as follows.

The first example is taken from the healthcare field. The sentence "`2.5 mg Albuterol` may be used to treat `acute exacerbations`, particularly in `children`." expresses a relation with 4 participants (`Albuterol`, `acute exacerbations`, `2.5 mg`, `children`) to accurately describe a treatment fact [1]. This kind of

The prime minister of Canada said: " The GDP in China was $13.41 trillion in 2018, $7.08 trillion less compared with the U.S., $1.17 trillion more compared with 2017."

Expressed facts:

1. (China GDP in 2018) = $13.41 trillion
2. (U.S. GDP in 2018) - (China GDP in 2018) = $7.08 trillion
3. (China GDP in 2018) - (China GDP in 2017) = $1.17 trillion

Relations:

R_{C8} : Economic-Index(China, GDP, 2018)
R_{C8e} : Equal(R_{C8}, $12.24 trillion)
R_{U8} : Economic-Index(U.S., GDP, 2018)
R_{U-C} : Subtract(R_{U8}, R_{C8})
R_{U-Ce}: Equal(R_{U-C}, $7.08 trillion)
R_{C7} : Economic-Index(China, GDP, 2017)
R_{8-7} : Subtract(R_{C8} - R_{C7})
R_{8-7e}: Equal(R_{8-7}, $1.17 trillion)

Fig. 1 Example of nested relations

Sentence 1.
The total revenue during 2015^1 and 2016^1 were $2 M and $2.2 M, increased 6% and 10% in 2015^2 and 2016^2, and the total cost were $1 M and $1.3 M.

Mention-*sensitive* (MS) annotation:
R_1 : Financial-Index(total cost, 2015^1, $1 M) or
R_1': Financial-Index(total cost, 2015^2, $1 M)

Mention-*insensitive* (MI) annotation:
R_* : Financial-Index(total cost, 2015 , $1 M)
ignores whether 2015 is 2015^1 or 2015^2

Sentence 2.
The total revenue during 2015^1 and 2016^1 were $2 M and $2.2 M, increased 10% in 2016^2, and the total cost were $1 M and $1.3 M.

Fig. 2 A sentence with mention repetition could be ambiguous for mention-sensitive annotation.

information appears in a vast number of medical articles. Extracting the complete relations is the foundation of large scale knowledge base construction in this field.

The second example shown in Figure 1 expresses relations about economic statistics. To describe the first fact, we first need a high-arity relation among three entities R_{C8}: Economic-Index(China, GDP, 2018), where Economic-Index is the relation type. We say relation R_{C8} has arity 3 (3 operands). In other words, it is a 3-ary relation. Then, we need a nested relation R_{C8e}: Equal (R_{C8}, $13.41 trillion). Note that although R_{C8e} is a 2-ary (binary) relation, its first operand is another relation R_{C8}. This is different from flat relations whose operands are restricted to be entities. R_{U-C}, R_{U-Ce}, R_{8-7}, R_{8-7e} are all nested relations. This kind of expression may come from politicians who want to convey his/her opinion on certain issues. As we are in a society congested with false-hoods and hyperboles, there are a lot of works on fact-checking of political discourses to ferret out misinformation [2]. Accurate and complete structured relation extraction from those claims is a vital component to this end.

Although high-arity and nested have equivalent expression power, high-arity formulation might have the problem of candidates explosion when the relation structure becomes complex and diverse (detailed in Section 3.2). Hence, we focus on nested relation extraction in this paper.

First, we give a formal formulation of nested relation extraction. It is a Directed Acyclic Graph (DAG) structure extraction problem, with an arbitrary number of layers and roots. Then, an Iterative Neural Network is proposed to extract nested relations layer by layer. It extracts one layer of relations at a time by generating all possible candidates and classifying them. The positive candidates become new relations to generate candidates in the next layer. This process iterates until no new candidates can be generated. The neural network has two major parts: 1) the horizontal part includes Bidirectional-LSTM and attention mechanism to represent entities, and 2) the vertical part is a relation representation neural network composed of DAG-LSTM which will grow as we extract more nested relations.

Furthermore, we observe that nested relations are usually expressed in long sentences where an entity concept may be mentioned repetitively. Since our model requires to specify the exact mention involved in the relation, this raises an issue on *entity mention selection* for annotation. Sentence 1 in Figure 2 illustrates such an example. Conceptually, we understand that sentence 1 expresses that the total cost in 2015 was $1M. That is

$$R_* = \text{Financial(total cost, 2015, \$1M)}.$$

However, 2015 is mentioned twice in previous clauses. Usually, we have to specify the exact mention of 2015 used in this relation during annotation. That is to say, annotators have to judge which one of R_1 and R_1' (in Figure 2) is correct. According to our experiment, random selection over these mentions will result in a poor performance (detailed in Section 6.3). So, for higher accuracy, it is required that a consistent annotation rule be used.

There might be two ways to design this annotation rule: mechanical and linguistic. Mechanical rules are ones that do not consider the language grammar, but just the mention positions. For example, the rule "use the nearest time mention occurred before the financial index" will assign R_1' as correct since 2015^2 is closer to total cost. However, management over a large number of these rules is difficult. What is worse, the annotation results from these

rules are not robust to the minor change of natural language expressions. For example, sentence 2 only changes the underlined part in sentence 1. But we have to choose 2015^1 in sentence 2, which is a quite different result with sentence 1 (where 2015^1 is regarded as not involved in the relation). Our experiments also demonstrate that the training data from these mechanical rules greatly deteriorate the model accuracy (Section 6.3).

Linguistic rules choose the mention that is more linguistically coherent. For the example in Figure 2, we choose R_1. The omitted 2015 in the last clause should refer to the mention in the first clause because the first and the last clauses are parallel structures, where the second clause is just an additional discussion of the "total revenue" in the first clause. Our experiments show that the dataset derived from a consistent linguistic rule for annotation gives the best model performance (Section 6.3). However, training in-house annotators to follow a consistent linguistic rule is costly, and it becomes more difficult if the annotation is out-sourced.

Therefore, we need a model that is not sensitive to the selection of entity mentions in the training data. That is to say, mention-insensitive models can use the training data where entity mentions in the involved relations are not specified. Specifically, we propose two ways to extend our iterative model. The first way is to aggregate repetitive entity mentions and give a synthesized representation of each entity concept. The second way is to first generate all possible relations over each entity mention, and then synthesize a representation over these relation candidates.

To evaluate the mention-sensitive iterative neural network model, two example tasks containing a large number of nested relations were introduced in the experiment. They are annotated by experts in mention-sensitive mode following linguistic rules. The semantic causality relation extraction task extracts semantic cause-and-effect relations. The formula extraction task extracts financial indexes and arithmetic relations. This task involves highly nested relations with diverse structures. The results show the effectiveness of our model.

To evaluate the mention-insensitive (MI) models, the mention-sensitive (MS) formula dataset is transformed into MI mode for training and evaluation. The loss of mention selection information pulls down the performances of MI models slightly compared with MS models. However, MI models are insensitive to the noise on mention selection, while the performance of MS models deteriorates quickly when the noise increases. The result shows that the MI models can achieve better results than MS models when the level of randomness on mention selection is higher than 0.3.

Our contribution is summarized as follows:

- To the best of our knowledge, this is the first study that gives a formal definition of nested relation extraction. We formulate nested relation extraction as a Directed Acyclic Graph (DAG) structure prediction problem.
- We propose a novel iterative neural network that can extract the nested relations layer by layer iteratively
- To alleviate the mention-selection problem, we propose two mention insensitive model on entity and relation level
- We empirically show the effectiveness of the proposed method by three experiments, and discover the "error propagation" phenomenon in nested relation extraction.

2 Related Work

Most of the RE researches extract relations between two entities [3–5]. Their research interests focus on model structure, joint training with named entity recognition, and external information incorporation [5–8]. Some new novel ideas include applying pointer network [9], walk-based method on entity graph [10], leveraging inter-sentence relation paths [11], etc.

Researchers have noticed the information loss of binary relation extraction during constructing knowledge graphs and ontologies like YAGO [12], so they attached temporal, geospatial and other prepositional information to relations. In the bioinformatics area [13], McDonald et al. [14] proposed an algorithm for high-arity relation extraction by discovering binary relation cliques. Ernst et al. [1] proposed a system to harvest high-arity relationships by pattern matching and bootstrapping. This phenomenon indicates the demand and necessity of complex relations for more precise information acquisition. To our best knowledge, no generic and systematic solutions about nested relation extraction were proposed.

While most RE datasets and models focus on mention-sensitive RE, the Biocreative V chemical disease relation extraction (CDR) dataset [15] annotated relations on the entity concept level. Thus, researches [16, 17] on this dataset usually first align the concept to mentions, then extract relation upon mentions. Verga et al. [17] proposed a model that computes all-pairs mention scores to perform multi-instance learning. By joint training with named entity recognition and aggregating over mentions to form entity pair scores, this model achieved the state-of-the-art result on

the CDR dataset. However, as the aggregation of entity pair scores is designed for flat binary relations, it is not suitable for nested relation extraction. The second MI model we propose shares the same insight of aggregating relations. But as we aggregate their representations instead of scores, they can be input into higher layers for nested RE.

Other related tasks including open information extraction, event extraction, and semantic parsing. Open information extraction (OIE) is a special kind of relation extraction that uses the predicates or text span from the text as relation type instead of predefined schemas [18–20]. Sun et al. [21] proposed a Seq2Seq model for open OIE which we use as a baseline in our experiment. Event extraction extracts a trigger word that indicates the occurrence of an event, then extracts its arguments like time, subject, etc [22]. Relation extraction techniques can be applied to the argument extraction. Semantic parsing is a wide concept [23] which covers works that map a natural language utterance into a formal representation like Lisp [24] or logical expressions [25, 26], database queries [23, 24, 27], and UCCA structures [28]. Most of them utilized Seq2Seq models, or transition-based models, which had to serialize the output structure into a sequence. But that may suffer from the situation that the generated sequences can be invalid [26]. Our proposed solution which directly generates a structure might provide a new approach in this field.

The DAG or Tree-LSTM model in our model has been applied in sentiment analysis [29], semantic relatedness classification [30], and traditional binary flat relation extraction [5]. These works used Tree-LSTM on parsed dependency trees as input to integrate external knowledge and get a better representation of sentence or relation pair. However, in our task, instead of applying Tree-LSTM on a given static structure, we use it to extract the task-specific nested structures.

3 Nested Relation Extraction

3.1 Problem Formulation

Nested Relation Extraction is the task of extracting semantic relations among multiple elements (entities or other relations), given a text and typed entities in the text.

A relation extraction task is usually carried out following a predefined task-specific schema. For example, the ACE 2004 relation extraction task [31] defined five general types of relations and further subdivided into a total of 24 types/subtypes of relations. Only semantic relations fall into these categories will be annotated for training and extracted during inference. For the example shown in Figure 1, one task may aim to extract all the relations shown in the figure, while another task only needs to extract the first fact and neglect the other two. Thus, defining the schema of a task is important in real-world problems, since this is the guide for annotation and model construction.

The schema of traditional binary relations between two entities is relatively simple. Here, we give a general framework to define the schema for a nested RE task by introducing notions of *entity*, *relation*, and *configuration* of relations. For a specific nested RE task, we need to define the task schema at the beginning to formally define what kind of relations under which constraints to extract (but semantically what is a relation is not defined here).

Entity. An entity mention consists of a word sequence and an entity label (type), denoted as $l(w_i, ..., w_j)$. It is a mention of a certain real-world entity. The example shown in Figure 1 contains entities like:

- `Index(GDP)`,
- `Country(China)`,
- `Value($, 13.41, trillion)`

where `Index`, `Country`, `Value` are labels and `GDP`, `China`, `$13.41 trillion` are words in entities. The entity types are omitted in the notation of relations in Figure 1. We denote the set of all kinds of entity labels in this task as E.

Note that this definition where entities are typed is in line with tasks like ACE 2004 and ACE 2005 [32]. But it is different from SemEval-2007 Task 4 and SemEval-2010 Task 8 in which all entities are "nominals" [33, 34]. Attaching labels to entities has advantages and disadvantages. The advantage is that we can shrink the search space and reduce the number of classification by constraining what kind of combination of entities may form a relation. The disadvantage is that we shift some of the burdens to the preceding named entity recognition module, and errors from that module may propagate. In this paper, we attach labels to entities in order to automate the extraction process (introduced in section 4).

Relation. We use *element* to refer to a relation or an entity. One relation consists of a tuple of elements and a relation label (the type of the relation). The behavior of one type of relations is defined as follows. Suppose l is a label, L_i is a set of labels, R^l the set of elements with label l, and $R^{L_i} = \bigcup_{l \in L_i} R^l$. R^l satisfies

$$R^l \subseteq R^{L_1} \times \cdots \times R^{L_k}$$

which means R^l is a subset of the Cartesian product of R^{L_i}. We call Cartesian product $L_1 \times \cdots \times L_k$ as the *configuration* of R^l, denoted as C^l. The behavior of relations with label l is defined by C^l. We denote L as the set of all the relation labels in this task.

A tuple of elements $(e_1, ..., e_k) \in R^l$, or has l relation, if and only if two conditions are satisfied:

1. this tuple satisfies C^l: $e_i \in R^{L_i}$ for all $i = 1, ..., k$, and
2. the l relation among these elements is semantically described in the sentence.

We denote this relation instance as $l(e_1, ..., e_k)$, and we call e_i the i-th operand of the relation.

For the example shown in Figure 1, the configuration of the `Economic-Index (EI)` relation is {`Country, Region`} × {`Index`} × {`Year, Quarter`}. That means the first operand of an `EI` relation must be a `Country` or `Region` entity, the second operand must be an `Index` entity, and the third operand must be a `Year` or `Quarter` entity. The formal notation of $\mathrm{R_{C8}}$ should be `EI(Country(China), Index(GDP), Year(2018))` where each of its operands is a typed entity instead of words. According to condition (1), tuples not satisfying the configuration, like `(China, U.S., 2018)`, will never have `EI` relation. `(Canada, GDP, 2018)` satisfying the configuration of `EI`. However, according to condition (2), since it is not semantically described in the sentence, this tuple does not have `EI` relation. This is why R^l is a subset of $R^{L_1} \times \cdots \times R^{L_k}$.

Schema. A schema of a nested RE task is defined by specifying the entity label set E, the relation label set L and the configuration of each relation $C^l, l \in L$.

Take the economic performance extraction task shown in Figure 1 as an example. The entity label set E, relation label set L, and configurations of relations are as follows:

$$
\begin{aligned}
E &= \{\texttt{Country}, \texttt{Region}, \texttt{Index}, \texttt{Year}, \texttt{Quarter}, \texttt{Value}\} \\
L &= \{\texttt{Economic-Index(EI)}, +, -, \div, =\} \\
C^{\texttt{EI}} &= \{\texttt{Country}, \texttt{Region}\} \times \{\texttt{Index}\} \times \{\texttt{Year}, \texttt{Quarter}\} \\
C^{-} &= \{\texttt{EI}, \texttt{Value}, +, -, \div\} \times \{\texttt{EI}, \texttt{Value}, +, -, \div\} \\
C^{+} &= C^{\div} = C^{=} = C^{-}
\end{aligned}
$$

When the configurations of some relations contain other relations, the relation structure can become nested. − and ÷ relations are mutually included, so the relations can be arbitrarily nested in this task.

3.2 Some Possible Solutions

We might convert nested relations into high-arity relations for extraction. High-arity and nested are different aspects of complexity as exemplified in introduction, and the same fact can be represented by both of them. The ternary $\mathrm{R_{C8}}$ relation in Figure 1 may be decomposed into two binary nested relations: `((China, GDP), 2018)`. On the other hand, facts expressed in nested relations can be transformed into flat high-arity relations: $\mathrm{R_{C8}}$ and $\mathrm{R_{C8e}}$ can be combined into a 4-ary relation `Economic-Index-Value(China, GDP, 2018, $12.24 trillion)`.

Although they have equivalent expression power, high-arity relations have the problem of explosion of candidates when the operands of a relation increase. For example, if we want to extract the third fact in Figure 1 using high-arity extraction, it involves 7 operands. As there are 3 countries, 1 index, two times and 3 values in the sentence, the number of operand combinations is $3 \times 1 \times 2 \times 3 \times 1 \times 2 \times 3$=108. But if we use nested relation extraction, we only have 48 candidates, since we prone many false combinations from lower layers. The difference between these two numbers increases fast if there are more operands in a high-arity relation and more entities in a sentence. In Section 6.2, we show this problem with a real-world task.

Seq2Seq model is a general approach for many NLP tasks. It has been used in related problems like open relation extraction [21] and word problem [35,36]. Take the sentence in Figure 1 as an example, a Seq2Seq model takes as input the sentence token sequence, and directly outputs the expressed facts as a sequence of tokens: "(China GDP 2018) = $13.41 trillion;(U.S. GDP 2018)-(China GDP 2018)=$7.08 trillion;(China GDP 2018)-(China GDP 2017) = $1.17 trillion". Obviously, this kind of formulation ignores the structure in the fact, thus has difficulty to achieve high performance on real-world problems. We compare the result of a Seq2Seq model in our experiment.

4 Solution with Iterative Neural Network

In this section, we introduce an Iterative Neural Network for nested relation extraction in Mention-Sensitive mode.

The major challenge of nested RE is that we cannot determine the total number of candidate tuples in a sentence, even though we know the number of entities. This is because relations can be arbitrarily nested. The traditional binary RE model can enumerate the tuples of entities (candidates) to classify. Thus, it can only extract the lowest layer of nested relations. High-arity RE models face with the explosion of candidates and Seq2Seq models ignore the structure information as we discuss in the previous section. So, we

propose to generate candidate tuples for classification on the fly. When new relations are extracted, they are used to generate the next layer of candidates. Therefore, the layer by layer process will extract nested relations of arbitrary layers.

During the extraction process, to classify a candidate, some RE methods generate features based on the left and right context of two entities [37]. They are not applicable to nested situations where the operand of a relation could be another relation. A nested relation could involve an arbitrary number of entities, and defining the context of multiple entities is complicated. Fortunately, deep learning is good at representation. Existing methods generate a vector of relation for classification. That vector also represents the semantic meaning of a relation. We represent relations and entities by using unified distributed representations, and classify them based on these representations. Thus, we are able to extract nested relations layer by layer iteratively.

4.1 Extraction Process Overview

In this subsection, we give an overview of the process that iteratively generates and classifies candidates.

We define a *candidate* with label l as a tuple of elements $(e_1, ..., e_k)$ that satisfies condition 1 of relation l, but whether its semantic meaning is correct or not (condition 2) is to be determined. We denote it as $l \leftarrow (e_1, ..., e_k)$. A left arrow is used to distinguish it with the notation of relation.

The overall process is to extract relations layer by layer iteratively. Steps to extract one layer of relations include:

1. generate candidates according to configurations,
2. classify them,
3. set the positive candidates as relations.

This process terminates when no new candidates can be generated in step 1.

In step 1, we generate all the candidates that can be generated (but not generated in previous layers) from the current state. At the beginning of the i-th layer, we put all entities and relations extracted from previous layers into a set R, all candidates generated before into a set Q. The candidates we need to generate form a set:

$$\{ l \leftarrow (e_1, ..., e_{k_l}) \mid l \in L \wedge e_i \in R \wedge (e_1, ..., e_{k_l}) \text{ satisfies } C^l \wedge l \leftarrow (e_1, ..., e_{k_l}) \notin Q\}$$

In step 2, to classify candidates, we propose an iterative neural network discussed in the next subsection.

4.2 Model Structure

The model is composed of three major modules. In horizontal orientation, there are text representation module and entity representation module. In vertical orientation, there is relation representation module. The relation representation module is suitable for the iterative generating and classifying process as its structure is flexible.

The iterative neural network model is illustrated in Figure 3. We use the example in Figure 1 to show the model structure. The words are presented at the bottom, followed by a text representation model composed of an embedding layer and a Bidirectional-LSTM. Then each entity is represented using attention mechanism over its tokens. Finally, in the relation representation model, hidden vectors of relations and candidates are computed layer by layer using DAG-LSTM cells. Hidden vectors of candidates are fed into fully connected networks for classification (shown as "cls" boxes). The positive ones (with checkmarks) might connect to the next layer. For example, the "-" cell is fed into the next layer because it is the operand of other candidates. However, "=" cell is positive but not fed into the next layer because it is not operand of any other candidates according to the task schema discussed in Section 3. The negative ones (with cross marks) are discarded.

The text representation module consists of a word embedding layer and a bi-directional LSTM [38] layer. This module converts each word into a hidden vector h_i that encodes information about this word and its context.

The embedding layer has an embedding look up table, which is a large matrix. The i-th vector in the matrix is the embedding of the i-th word in the vocabulary.

The uni-directional LSTM network is a variant of the recurrent neural network. It takes as input a token sequence. At each step (token), an LSTM cell produces a new hidden vector by combining previous step cell and hidden and the current step input vector:

$$h_t = f(h_{t-1}, c_{t-1}, x_t).$$

The hidden vector encodes information of the input sequence from the beginning to the current step. Bi-directional LSTM contains two uni-directional LSTMs (forward and backward). The hidden vector of each word is the concatenation of two LSTM outputs.

The entity mention representation module is used to represent entity mentions. One mention might consist of multiple words and thus multiple hidden vectors. However, a fixed-length representation is wanted for a mention. So we

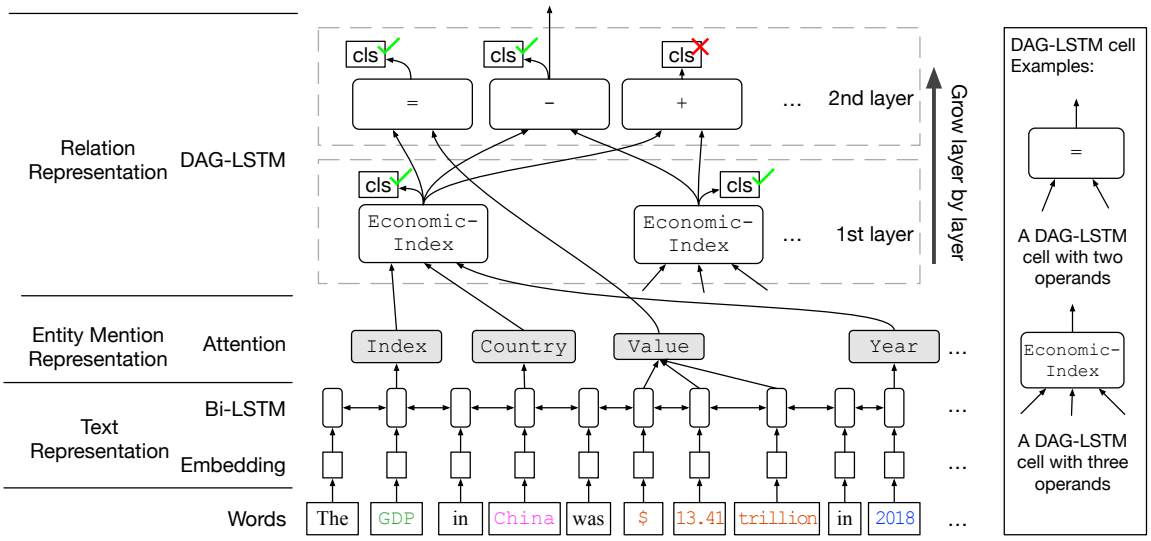


Fig. 3 Model structure with an input example

apply a dot product attention [39] on words in this mention. Suppose the word vectors in this mention are $(h_1, h_2, ..., h_n)$, the hidden state h of mention is computed as:

$$\begin{aligned} s_i &= w^T h_i, \text{ for } i = 1, ..., n \\ a_i &= \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \\ h &= \sum_{i=1}^{n} a_i h_i \end{aligned} \tag{1}$$

where w is a trainable parameter.

The relation representation module applies DAG-LSTM to represent relations and candidates. The DAG-LSTM is an extension of Tree-LSTM [30, 40] that has been applied in many tasks [5, 30]. One DAG-LSTM cell combines hidden vectors of multiple operands into a vector to represent its corresponding relation or candidate. Multiple layers of DAG-LSTM cells will let the information flow among the nested relations and candidates. One DAG-LSTM cell takes as input the hidden and cell states h_j, c_j from each of its operands, and the embedding e of the relation label. For simplicity, we concatenate hidden vectors into vector $v = [h_1, ..., h_k]$. The computation process is as follows:

$$\begin{aligned} i &= \sigma(W^i e + U^i v + b^i) \\ f_j &= \sigma(W^f_j e + U^f_j v + b^f_j), \quad j = 1, ..., k \\ o &= \sigma(W^o e + U^o v + b^o) \\ \hat{c} &= \tanh\left(W^c e + U^c v + b^c\right) \\ c &= i \odot \hat{c} + \textstyle\sum_{j=1}^{k} f_j \odot c_j \\ h &= \tanh(c) \odot o \end{aligned} \tag{2}$$

where W^*, U^*, b^* are network parameters, $\odot$ is the element-wise product, h and c are hidden and cell states of this relation or candidate, i, f_*, o are input, forget, and output gates as sequential LSTM [38]. To classify a candidate, we feed its hidden vector h into a multi-layer fully connected network followed by Softmax function to compute its probability of correctness.

During training, the annotated data only contain correct relations, so we generate all possible candidates. Candidates appear in annotation results are positive samples, and others are negative samples. One sentence contains many samples (candidates), we compute though text and entity representation module for one time, which output the hidden vectors of all entities. Then all candidates in this sentence are computed based on these vectors. This will reduce redundant computations compared to re-compute entity hidden vectors for each candidate.

The training objective is to minimize the negative log-likelihood over all candidates:

$$L = \sum_{s \in D} \sum_{c \in C(s)} y_c \log(p(c)) + (1 - y_c) \log(1 - p(c))$$

where D is a dataset, s is one sentence, $C(s)$ is the set of all candidates in s, y_c is the label of candidate c, and $p(c)$ is the probability our model predicts that c is a relation.

5 Mention-Insensitive Extraction

The iterative neural network we introduced above is a mention-sensitive (MS) model that requires to specify the

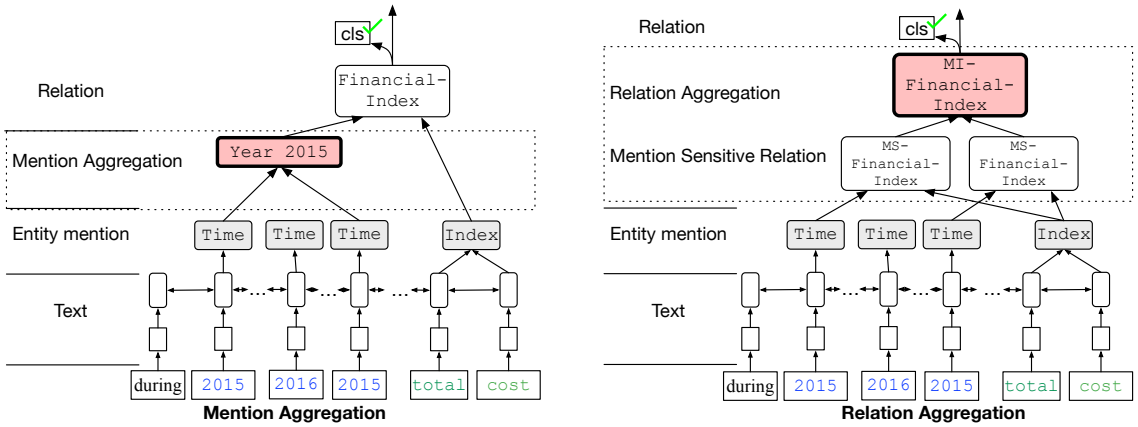


Fig. 4 Model in mention-insensitive mode. Nodes with thick edges are aggregation nodes.

exact mention involved in relations. In this section, we extend the iterative model from two aspects to adapt it to the mention-insensitive (MI) mode that only requires to specify the entity concept involved in relations.

5.1 Mention aggregation

The first way to extend the iterative model to MI mode is by mention aggregation.

The *mention set* $M(e)$ of entity concept e is the set of all mentions of e in the sentence. For example, in sentence 1 in Figure 2, mention set of `2015` is $M(\texttt{2015}) = \{\texttt{2015}^1, \texttt{2015}^2\}$. So, both $\texttt{R}_1$ and $\texttt{R}'_1$ will be `Financial-Index (total cost, M(2015),$1M)` in MI mode.

Corresponding to the mention set concept, we add an extra "mention aggregation" layer between entity mention and relation representation layer to our model as shown on the left side of Figure 4. This layer aggregates mentions in the same mention set. The aggregation method we adopt is the attention mechanism. Given a mention set $M(e) = \{m_1, m_2, ..., m_n\}$ of concept e, each mention m_i has a representation h_i from the entity mention module. In the mention aggregation module, the representation h_e of $M(e)$ is computed using the same attention model as Equation (1) in the entity mention module. Then, h_e, as the representation of all the mentions of e, is fed into the relation representation module.

The overall process of layer by layer iterative extraction is not changed. But when generating candidates, we use mention set instead of individual mention if the mention is repetitive.

5.2 Relation aggregation

Aggregating entity mentions does not model the interaction between entity mentions. The mentions of an entity are mixed up before fed to involved relations. To model the interaction between mentions, we propose the relation aggregation model that first represents the MS relations, and then aggregates them (shown on the right side of Figure 4).

For a relation type l that may have repetitive mentions in operands, we split it into two relation types `MS-`l and `MI-`l. `MS-`l inherits the configuration of l. `MI-`l's configuration is special: it may have a flexible number of `MS-`l relations as operands.

The MI mode annotates upon the entity concept. Thus, for relation $r = l(M_1, ..., M_k)$, we define the *MS relation set* of r as:

$$S(r) = \{\texttt{MS-}l(s) \mid s \in M_1 \times M_2 \times ..., \times M_k\}$$

where $\times$ denotes the Cartesian Product. Each relation in $S(r)$ is mention insensitive. But whether or not each of them is correct is unknown in MI annotation.

Then, we define an *aggregation relation* $\texttt{MI-}l(S(r))$ which takes the MS relation set of r as operands and `MI-`l as relation type. In summary, a mention-insensitively annotated relation will be transformed as:

$$r = l(M_1, ..., M_k) \quad \rightarrow \quad S(r),\ \texttt{MI-}l(S(r)).$$

For example, the relation $\texttt{R}_*$ shown in Figure 2 is transformed into three nodes in Figure 4. The MS relation set $S(\texttt{R}_*)$ has two nodes with label `MS-Financial-Index`, and the aggregation relation is the `MI-Financial-Index` node with a thick edge.

Consequently, in the model, we first represent each relation in $S(r)$, then aggregate them to represent

$MI - l(S(r))$. This aggregation is also achieved by attention mechanism the same as the entity mention representation module. Such design allows each MS relation to represent a concrete relation among mentions and the aggregation relation chooses some good MS relations as its representation by attention model.

This relation aggregation model takes different steps in each layer of the iterative extraction process. The 1) generate candidate, 2) classify and 3) select relation steps of each layer are changed into the following steps:

1. generate `MS`$-l$ relations according to C^l,
2. generate `MI`$-l$ candidates by a special generation procedure that aggregates MS relation set,
3. classify `MI`$-l$ candidates
4. set the positive `MI`$-l$ candidates as relations.

where the original generating candidate step becomes the first 2 steps. The training loss only contains the classification loss of MI relations, as the labels of MS relations are unknown. Other parts of the overall process are the same as the iterative neural network.

6 Experiment

Three experiments are conducted in this section. The first two experiments extract nested cause-and-effect relations and formula relations to show the effectiveness of our mention-sensitive iterative neural network on nested RE. The third experiment evaluates our mention-insensitive models, comparing with mention-sensitive models trained on different levels of noise.

The hyper-parameter settings of our model are as follows. The vocabulary size is 6000 because the vocabulary in the financial dataset is relatively concentrated and we transform time, financial index phrases into special tokens. The embedding size and hidden vector size of Bi-LSTM are set to 256. We add position encodings to word embeddings following [41]. Since Bi-LSTM will concatenate hidden vectors from forward and backward, we set the hidden size of DAG-LSTM to 512. The final fully connected layer is $512 \times 1024 \times 2$. We use Adadelta [42] as the optimizer with learning rate = 1.0 and batch size = 8. The dataset is split into 6:2:2 for training, validation, and testing. We apply the early stop technique and use the best model on the validation set for evaluation.

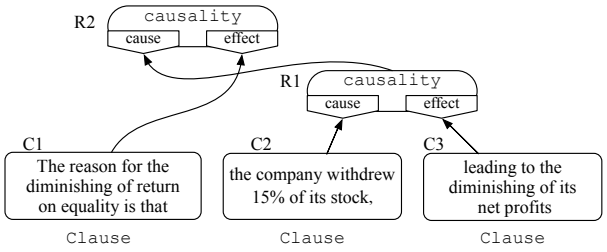


Fig. 5 An example of semantic causality extraction.

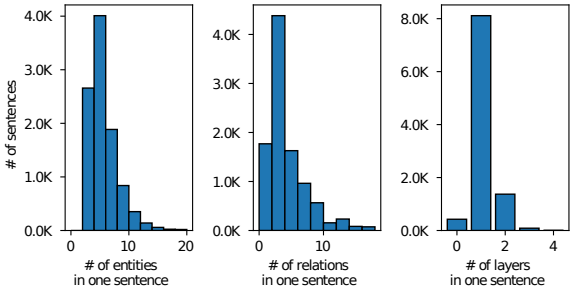


Fig. 6 Distributions of sentences in semantic causality dataset.

The evaluation metrics are defined as

$$\text{Precision} = \frac{\sum_{s \in D} |\hat{R}_s \cap R_s|}{\sum_{s \in D} |\hat{R}_s|}$$

$$\text{Recall} = \frac{\sum_{s \in D} |\hat{R}_s \cap R|}{\sum_{s \in D} |R_s|}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where s is a sentence from dataset D, $\hat{R}_s$ is the set of predicted relations in s, and R_s is the set of annotated relations in s. Two relations are the same if both their operands and labels are the same.

In the first two experiments, we analyze our model in two modes: "**From scratch**" means that the model predicts relations layer by layer, and errors in one layer will propagate to the subsequent layers. This is the situation when the model is applied in the real world. "**Guided**" means that the prediction results of the current layer are replaced by correct results when predicting the following layers. This is the situation when we are training the model. We find that there is a performance gap between these two modes (detailed in the experiments).

6.1 Semantic Causality Relation Extraction

Task Description. Semantic causality relation is the relation that describes cause and effect between clauses or relations. For example, Figure 5 shows a sentence with three clauses

Table 1 Statistics of two datasets. "#nested relation" means the number of relations whose operands contain other relation(s).

	#sentence	#relation	#nested rel	#root	#relations in layer				
					1	2	3	4	⩾ 5
Formula	106,048	876,493	484,501	348,349	41.19	43.27	14.49	0.55	0.50
Causality	10,000	39,812	6,572	36,417	83.49	15.44	1.04	0.03	0.00

C1, C2, and C3. C2 is the cause of C3, which forms a relation R1. R1 (or the combination of C2 and C3) is the cause of C1. More examples are shown in appendix. The task schema is:

$$\begin{aligned} E &= \{\texttt{Clause}\}, \\ L &= \{\texttt{Causality}\}, \\ C^{\texttt{Causality}} &= \{\texttt{Clause}, \texttt{Causality}\} \times \\ &\quad \{\texttt{Clause}, \texttt{Causality}\} \end{aligned}$$

Statistics. Ten thousand of sentences are selected and manually annotated from bond prospectuses in Chinese. The basic statistics are reported in Table 1. The distributions of the sentences are shown in Figure 6. There are 14.62% of sentences that have nested relations.

Baseline. At first glance, this problem seems to be simple due to the existence of keywords like "reason", "leading to", and "because". So we attempt to solve this problem by rule. By digging into the annotated data, 293 distinct keywords are summarized into 4 categories. Then, each sentence is converted into a pattern sequence consisting of clauses and keywords, and 1344 patterns are found. We summarize over 130 most frequent patterns based on these patterns and their relation results, and come up with an expression parsing-like algorithm. This algorithm can cover 518 patterns, more than 88% of sentences.

Since most of the sentences have 1 or 2 layers of relation, we convert nested relation like R2=causality(R1, C3) in Figure 5 into a ternary flat relation causality3(C1, C2, C3), and extract binary and ternary relations together using our model. Binary and ternary relations cover 99% of all the relations. We denote this method as "High-arity".

The Seq2Seq model is not suitable for semantic causality extraction because each entity is a clause. Sequences like "((A cause B) cause C); E cause F", where A, B, C are clauses, is too long and too difficult to predict.

Result Analysis. The results on the test set are shown in Table 2. Besides the overall result, in Figure 7, we also report the performances on each layer. Since the relations in layer 3 or higher are rare (0.7%), we report the result of relations in layer 1, and relations in layers ⩾ 2.

First, we compare the results between the rule-based algorithm and our model (from scratch). The first row of the table shows that the performance of the rule-based algorithm is poor. The precision of the rule algorithm is 7% lower than our model. Although we have put efforts on rule designing and many rules are collected, the recall is still very low (< 18%). Designing more rules may increase its recall but may hurt the precision, and bring conflict among rules. The second row shows the result of our model. The overall F1 score is 81.08, which is much higher than the rule algorithm. In Figure 7, layer-wise evaluation shows a drop of performance of our model on the second layer, but still significantly outperforms hand-crafted rules. The comparison between rule and our model indicates that the semantic causality extraction is a non-trivial problem, despite the existence of strong causal conjunction features like "since" and "because".

Second, we compare the result of high-arity and our model. High-arity performs slightly worse than our model (77.49 vs. 78.51), and the difference comes from the recall on the second layer as shown in Figure 7. This means that

Table 2 Results on semantic causality extraction.

Model	Precision	Recall	F1
Rule-based	70.80	17.48	28.04
High-arity	77.49	79.50	78.48
Our model (From scratch)	78.51	79.41	78.96
Our model (Guided)	81.63	80.99	81.31

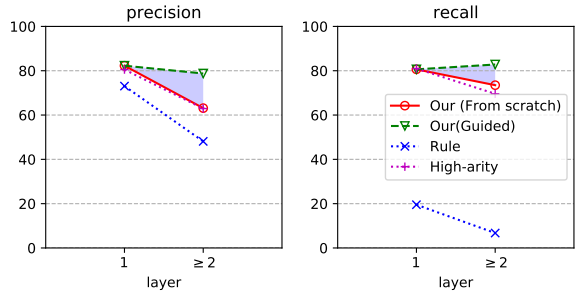


Fig. 7 Performances on different layers of relations on semantic causality extraction task. "Guided" means that at the i-th layer, we use the correct annotation result of previous layers to generate candidates., "From scratch" means no label information is used.

when the nested relations do not have many layers, nested formulation still outperforms flat high-arity formulation.

Third, we introduce the phenomenon of *error propagation*. In Figure 7, the performance of Guided is the same as From Scratch at layer one since their entities are the same, but is over 10% better than "From scratch" in layers $\geqslant 2$. That means the error in the first layer will propagate and accumulate through layers which brings in two types of errors. 1) If a relation is not recalled in the first layer, then relations in subsequent layers that take it as an operand can not be extracted. This will affect the recall of subsequent layers. 2) If a relation is falsely extracted, the process might generate strange candidates based on it which are never seen during training. This will affect the precision of subsequent layers.

We summarize three reasons for the decline of performance in higher layers: 1) extracting the high layer relations is intrinsically harder in semantic as its more complex, 2) there are fewer relations in high layers, and 3) error propagation and accumulation through layers.

6.2 Formula Extraction

Task Description. Verbal descriptions over the numerical relationships among some objective measures widely exist in financial documents. We collect a large manually annotated dataset from bond prospectuses.

Figure 8 shows an example from our dataset. Three types of entities are extracted in advance. A time entity describes a certain time like year 2016. An index entity describes a financial index, like "other payables". A value entity describes a value like $ 80 million or 98%. The first kind of relation is the relation between time and index. They are shown as relations with the "@" mark. This kind of relations refer to financial indexes at a certain time, which should be equal to some numbers. Then, the higher layers of relations are summations, divisions, and equations. More examples are shown in appendix. The task schema is:

$$
\begin{aligned}
E &= \{\texttt{Time, Index, Value}\} \\
L &= \{\texttt{@, +, -, ÷, =, ≈, <, CGR, Prop}\} \\
C^{@} &= \{\texttt{Time}\} \times \{\texttt{Index}\} \\
C^{+} &= \{\texttt{@, +, -, ÷, Value}\} \times \\
&\quad \{\texttt{@, +, -, ÷, Value}\} \\
C^{=} &= C^{\approx} = C^{<} = C^{-} = C^{\div} = C^{\texttt{CGR}} = C^{+} \\
C^{\texttt{Prop}} &= \{@, +, -, \div\}
\end{aligned}
$$

The `CGR` (compound growth rate) relation is the relation described in sentences like "the compound growth rate of income from 2005 to 2018". The `Prop` relation is a division where the divisor is omitted in the sentence, so it has only one operand. We convert all ">" relations into "<" relations by reversing their operands since we think they are syntactically similar in natural language. So, there are 9 types of relations in this task.

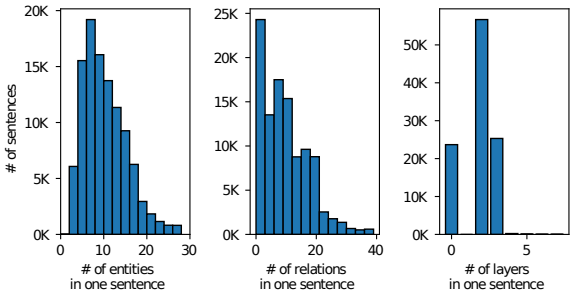


Fig. 9 Distributions of sentences in formula extraction dataset.

Statistics. More than a hundred thousand of sentences are annotated. The basic statistics are reported in Table 1. The distribution of the number of relations and layers in a sentence are shown in Figure 9. Note that there are sparks at the number of relation = 0. Since the sentences are not strictly filtered, there are many sentences expressing no relations. Also, note that no sentences have one layer of relations. The first layer of relations must be @ relations according to the definition above. Our annotation guide requires annotators to annotate @ relations that are operands (or descendants) of equation or comparison relations. If it is not operand of any other relations, it will not be annotated. So, if there are relations in a sentence, it must have at least two layers. Second layer relations are relations like = relations between @ relations and numbers, or the arithmetic relation between @ relations. Third layer relations are relations like = relations between arithmetic relations and numbers, or more complex arithmetic relations. The distribution shows that the relations are highly nested in this dataset: more than one-fourth of sentences have 3 or 4 layers of relations.

Baseline. High-arity relation extraction is infeasible for this task due to the explosion of the number of candidates. If we get each root of the nested structure and analyze its structure, we find there are 161 templates in test data. A template looks like T1=`Time@Attribute+Time@Attribute=Value`, can be regarded as a configuration. To cover 98% (the recall our model achieves in Guided mode) of the relations, we need the most frequent 11 templates. One of the top 11

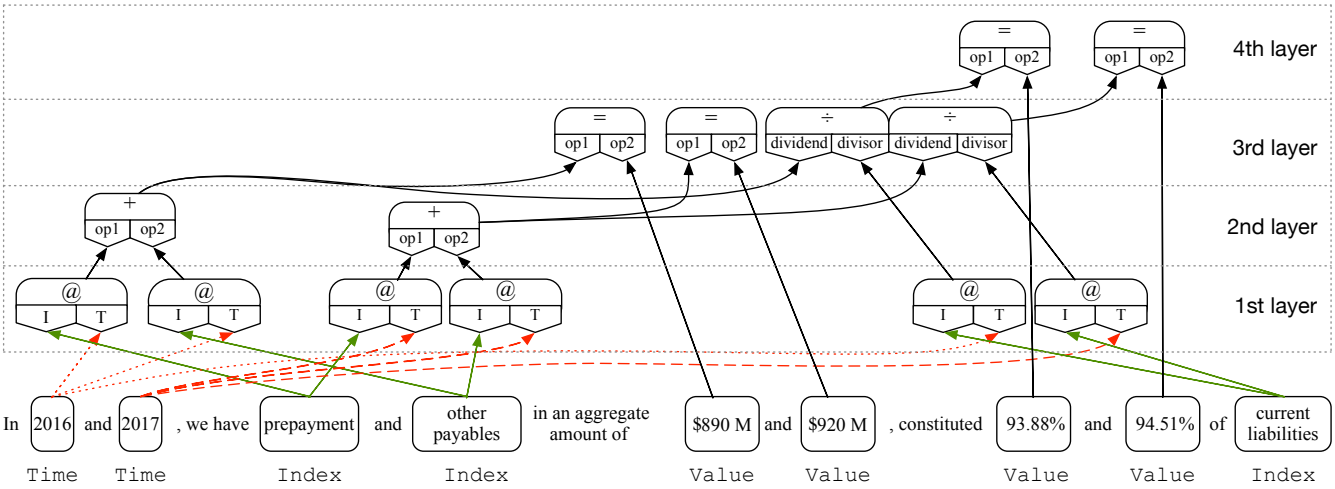


Fig. 8 Example of formula extraction.

Table 3 Results on formula extraction.

Model	Precision	Recall	F1
Seq2Seq	86.56	75.59	80.71
Our model (From scratch)	96.05	95.85	95.95
Our model (Guided)	97.97	97.99	97.98

templates involves 5 time entities. If a sentence has 10 time entities, there are 10^5 possible combinations of time entities for one template. As a result, the top 11 templates generate 1525 times more candidates than our nested approach. It is infeasible to directly extract flat high-arity relations due to the limitation of computing power.

For Seq2Seq model, we use the Seq2Seq model from [35, 43], and incorporate attention [41] and copy mechanisms [44] to improve its performance. The entities in the sentence are replaced by special tokens, like "In t_1 and t_2, we have i_1...", and the output is a sequence like $i_1@t_2 + i_2@t_2 = v_1$. If there are multiple formulas in a sentence, we concatenate them by a special token ";" and their order is determined by the position of elements in each expression. Moreover, following [36], we transform the expression from infix to prefix as a normalization method. As a result, a sentence with two formulas might output: "$=+@i_1t_2@i_2t_2v_1; =@i_3t_3v_3$". Our implementation is based on an open source neural translation system OpenNMT [45].

Result Analysis. The results on the test set are shown in Table 3 and the layer-wise performances are shown in Figure 10. The first intuition is that this result is better than the result of semantic causality. We think this largely attributes to the data size: there are 10 times more data in this task, and deep learning models are data-hungry.

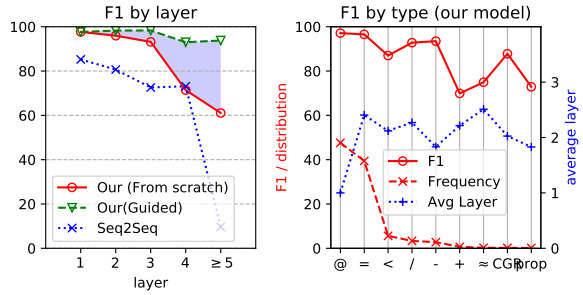


Fig. 10 Performances on different layers, and on different relation types on formula extraction task.

Our model significantly outperforms the Seq2Seq model, especially on recall and on higher layer relations. This is because there are too many relations in one sentence, and the orders of them are too difficult for Seq2Seq models to learn. Moreover, learning the structured sequence is hard for Seq2Seq models.

In Figure 10, the result of Guided on different layers are stable, F1 score drops slightly from 97.98% on layer 1 to 93.79% on layers $\geqslant 5$. That means our model is capable of extracting nested relations in high layers. Notice that, F1 score in layer 1 is lower than layer 2 and 3. The reason is as follows. Highly nested relations need lower layer relations as operands. As mentioned above, our annotation guide asks to annotate relations that are descendants of equation or comparison relations only. Some of the lower layer relations are very subtle because the model has to discover the high layer relations first before extracting them. For example, in the sentence "The income in 2019 was good.", although it mentions income@2019, as it is not used in other

comparison relation, it is not annotated in the dataset (to reduce the labeling effort). Thus extracting these lower layer relations is not simpler than high layer ones. On the other hand, when the lower relations are extracted, extracting higher layer relations becomes easier. The results on layers 4 and 5 are lower than layers 1-3 because the number of relations in layers 4 and 5 is one order of magnitude smaller, as shown in the "percentage" row.

The result of From scratch is the same as Guided on layer 1 since they have the same candidates. The result of From scratch has a sharp drop from layer 3 to layers 4. The gap between Guided and From scratch is over 20% on layer 4 and layers $\geqslant 5$. Thus, the sharp drop should largely attribute to the error propagation (illustrated by the large gap). Compared with the result of semantic causality extraction, the error propagate in this task is worse, as the number of layers is larger. There are researches on alleviating error propagation problem from named entity recognition to RE by joint training [5] and novel tagging [46]. How to alleviate error propagation in nested RE is an interesting future work.

In the right subfigure of Figure 10, we also report the performance by the type of relation, as well as the frequency and average layer number of each type. The performance is positively related to the frequency and negatively related to the average layer.

Efficiency. The average inference time per sentence is 0.042s on the formula extraction task, 0.024s on the causality extraction task. It is almost "realtime" to predict relations in one sentence, but it takes several minutes to process a large document.

6.3 Mention-Insensitive Formula Extraction

Task Description. In this subsection, we test our MI models on the transformed formula dataset and compare them with MS models. Mention repetition is rare in the cause-and-effect dataset, so we do not test on it.

The formula extraction dataset discussed above contains a huge number of entities, and the mention repeats frequently (like the example shown in Figure 2). We transform the formula dataset into an MI annotation result as follows. In this experiment, only mentions with the same expression are regarded as the same entity concept, and entity linking and coreference resolution are not used. That is because, in financial documents, wordings for the same entity concept are usually the same, and pronouns are used rarely. For other tasks where concepts have diverse expressions and pronouns are used frequently, entity coreference resolution would be a necessary preprocessing. We only regard the same expressions of `Time`, `Index` mentions as the same concepts, which means only "@" relation will be directly affected according to the task schema of formula extraction. `Value` mentions are rarely repeated and mentions like 100% may not have the same meaning when repeated.

The formula dataset is annotated by full-time annotators following linguistic rules. Every annotator gets strict training before start and takes daily meetings on the new difficult examples. Every sentence is annotated by two annotators and the conflicts are judged by another moderator. Such rigor procedure produces a high-quality MS dataset. However, as crowdsourcing is currently the dominating data collection paradigm, high quality MS annotation dataset like our formula dataset might be hard to obtain.

We imitate the result following mechanical rules. For a "@" relation, it has two operands: time and index. If one of its operands is repetitive, we make rules on how to choose one from the mention set. We consider the distance between operands, whether they are in the same clause, and the order they appear in the sentence. Among the combinations of these conditions, two kinds of rules make the least changes on the clean MS dataset:

- A. Find the closest time and index mention while trying to make the time mention appears before the index.
- B. If exist, use the time and index in the same clause. Otherwise, use rule A.

The details of rule A and rule B are shown in the Appendix.

We also imitate the inferior annotation result from crowdsourcing by adding noise to the MS dataset. For an operand of a relation, if it is an entity and its corresponding mention set $M(e)$ (refer to Section 5.1) has more than one element, a replacement action sets this operand with an arbitrary mention in $M(e)$. The probability α of taking a replacement action is called the random level in mention selection, which indicates the proportion of relations that are annotated by not well-trained annotators who choose mentions randomly.

Statistics. We denote the formula dataset as D^+, and sentences that contain repetitive mentions as D. $|D|/|D^+| = 31.70\%$ where $|D|$ is the number of sentences in D. As we concentrate on the mention-sensitivity problem, using D^+ for training and evaluation might dilute the effect of mention repetition phenomenon. Thus, D is used throughout this experiment. Two datasets are derived from D: D_{MS} and D_{MI} for the MS and MI annotation methods.

Table 4 F1 scores of Mention-Insensitive models and comparison to Mention-Sensitive model

	Model	Trained on	all relations 100.00%	rep. 33.11%	non-rep. 66.89%	@ 42.02%	@(r) 10.97%	@(n) 31.05%
MS-D_{MS}	Mention-Sensitive	D_{MS}	94.05	91.43	95.35	97.93	96.28	98.51
MS-$D_{MS}(A)$	Mention-Sensitive	$D_{MS}(A)$	87.90	84.63	89.51	93.98	91.62	94.80
MS-$D_{MS}(B)$	Mention-Sensitive	$D_{MS}(B)$	88.71	85.47	90.32	94.35	92.33	95.05
MI-m-D_{MI}	Mention Aggregation	D_{MI}	91.84	88.54	93.49	97.16	95.38	97.79
MI-r-D_{MI}	Relation Aggregation	D_{MI}	91.48	88.58	93.05	97.22	95.57	97.84

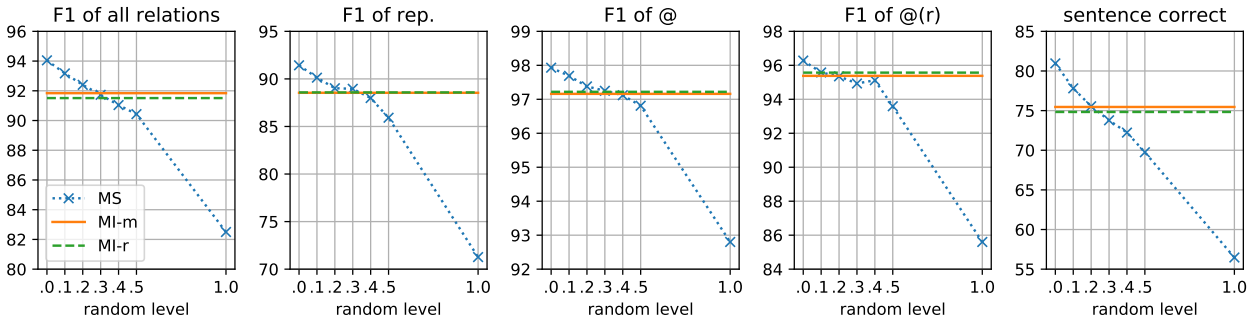


Fig. 11 Comparison of MI models with MS models under different levels of noise.

We denote the transformed MS dataset following rule A and rule B as $D_{MS}(A)$ and $D_{MS}(B)$. The percentages of corrupted relations to relations containing repetitive mentions are 21.76% and 20.72% respectively. The random level α is set to 0.1, 0.2, 0.3, 0.4, 0.5 and 1, and the corresponding datasets are denoted as $D_{MS}(\alpha)$. Note that a replacement action might choose the original correct mention, resulting in a null operation. Thus the actual changes occurred is less than α. The percentages of corrupted relations to relations containing repetitive mentions are 5.37%, 10.90%, 16.77%, 21.83%, 27.83%, and 55.18% respectively.

Result Analysis. Results of MS models are converted into MI results for evaluation so that all results are compared under MI mode. To show the results in more detail, models are evaluated on several subsets of relations:

- "all relations": all relations,
- "rep.": relations having repetitive mention (RM) descendants,
- "non-rep.": relations having no RM descendants,
- "@": @ relations (between `Index` and `Time`),
- "@(r)': @ relations having RM operands,
- '@(n)": @ relations having no RM operands,

where the descendants of a relation r are all the elements under the tree rooted at r in the DAG.

(1) MS-D_{MS} vs. MS-D^{+}_{MS}. In Table 4, on all relations, the F1 score of MS-D_{MS} drops about 2.0 points compared with the result of MS-D^{+}_{MS} (in the last experiment). This is because dataset D is smaller and more difficult than D^{+}.

(2) MS-D_{MS} vs. MS-$D_{MS}(A/B)$. In Table 4, compared with MS-D_{MS}, MS-$D_{MS}(A/B)$ F1 scores drop shapely (about 5.3 to 6.1). We examine their precisions and recalls, and find that the recall on @ relation drops dramatically from 98 to 90, while the precision keeps at 98. The reason is that the model might not precisely handle the mechanical rules which requires a precise count on repetitions and measure the distances. Instead, learning from mechanical rules, models discard many @ relations that are semantically correct (like $\mathtt{R}_1$ in Figure 2). Such result broadcasts from @(r) to @(n), severely harm the recall. As @ relations are the first layer relations, the relations on higher layers perform poorly due to error propagation.

Although the percentages of corrupted relations of $D_{MS}(A/B)$ are smaller than $D_{MS}(0.4)$, MS-$D_{MS}(A/B)$'s performance is worse than MS-$D_{MS}(0.4)$ (as shown in Figure 11). We think the reason is that $D_{MS}(0.4)$ is uniform-randomly corrupted, but $D_{MS}(A/B)$ is corrupted biased to a certain rule. Thus, the model might not learn the noise from $D_{MS}(0.4)$, and only can focus on the correct relations; while the model might partially fit the biased rules from $D_{MS}(A/B)$, which affect the ordinary relations. Thus, the MS model on datasets following mechanical rules

performs poorly on this task.

(3) MI-r/m-D_{MI} vs. MS-D_{MS}. In Table 4, due to the information loss on the mention selection, MI models perform poorer than MS-D_{MS}. The results of MI-m and MI-r are quite similar, dropping around 2.0 to 2.4 points. On "rep." and "non-rep.", all the three models perform poorer on rep. than non-rep. Performances of MI models drop more on rep. than non-rep. (2.9 vs. 2.1 on average), and similar for @(r) and @(n) (0.8 vs 0.7 on average).

The drop of MI models on rep. and @(r) is as expected since there is information loss on such relations. The reason that performance also drops on non-rep. and @(n) is as follows. In MS dataset, both rep. and non-rep. relations are annotated mention-sensitively. Thus all relations are homogeneous (mention-sensitive). But in MI dataset, rep. and non-rep. are heterogeneous: non-rep. relations know the exact mention (mention-sensitive) while rep. relations do not (mention-insensitive). So, the data used to learn the mention-sensitive relations (non-rep.) in D_{MI} is smaller than D_{MS}. Therefore, the performances on non-rep. and @(n) also drop. Study on how to keep the performance of non-rep. relations is an interesting future work.

(4) MI-r/m-D_{MI} vs. MS-$D_{MS}(\alpha)$. We try to understand the difference of performances between MI and MS models quantitatively. So, we compare MI models with MS models that trained on different random levels of mention selection. All of these models are tested on the same gold annotation. The result is shown in Figure 11. As MI models are insensitive to noise, their performances are flat w.r.t. random level. The first four subfigures are F1 scores on different subsets of relations. The right-most subfigure shows the percentage of sentences whose structures are extracted perfectly. From Figure 11 we can see that the performances of our MI models are close to MS-$D_{MS}(0.3)$. Thus, if a dataset is annotated with 30% or more randomness on relations with repetitive mentions, we can adapt MI models to extract nested relations for better performance.

7 Conclusion

In this paper, we extend the concept of relation extraction from flat relations to nested relations. We propose a formal formulation of the nested relation extraction problem and introduce an Iterative Neural Network (INN) that is able to extract nested relation layer by layer. This model performs well on two different nested relation extraction tasks.

Moreover, to handle the noise from mention-sensitive annotation, we extend INN from two aspects: mention aggregation and relation aggregation. These extensions will perform better compared with vanilla INN when the random level of mention selection is above 0.3.

Some future research directions include how to alleviate the error propagation problem and explore more applications of nested RE. How to train the model with less data is also important, since labeling complicated nested relation is time-consuming and is the largest barrier in applications.

8 Acknowledgements

This work was supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant No. U1811461, and the Innovation Program of Institute of Computing Technology, CAS.

A preliminary version of this work has been published in the Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM) [47].

Appendix

Table 5 More examples from two datasets, translated from Chinese.

Semantic Causality
1. (C1) The net income of company A was negative in 2016 (C2) because the operating costs were increasing, (C3) and the industry had been sluggish in recent years. $(C2 \rightarrow C1), (C3 \rightarrow C1)$
2. (C1) The net income was negative in 2016 (C2) because the projects under construction had not got the presale qualification, (C3) and therefore the income was not carried forward, (C4) leading to a deficit. $((C2 \rightarrow C3) \rightarrow C4) \rightarrow C1$
Formula (We omit simple relations for brevity.)
1. In 2015, the inventory was \$11.46M, constituted 5.64% of the total assets, decreased \$1.84M compared with 2014, and fell by 13.83%.
2. In years 2014, 2015, 2016 and 2017, the liquid liabilities were \$3,539M, \$2,651M, \$2,585M, and \$2,148M, constituted 16.89%, 11.87%, 11.22%, and 9.24% of the total liabilities, and the ratios were trending downward. (Here, "trending downward" implies 3 "<" relations among 4 ratios)

Algorithm 1 Mechanical Rules A and B

```
function ruleA(times, indexes)
    if multiple(times) and multiple(indexes) then
        times = preserve-the-first-one-in-sentence(times)
    end if
    if single(times) and multiple(indexes) then
        time = times[0]
        index = nearest-after(indexes, time)
    else if multiple(times) and single(indexes) then
        index = indexes[0]
        time = nearest-before(index, times)
    else
        time, index = times[0], indexes[0]
    end if
    return time, index
end function

function nearest-after(mention-list, pivot)
    mentions-after = preserve-after(mention-list, pivot)
    if exist(mentions-after) then
        mention-list = mentions-after
    end if
    return the-nearest(mention-list, pivot)
end function

function ruleB(times, indexes)
    if multiple(times) and multiple(indexes) then
        times = preserve-the-first-one-in-sentence(times)
    end if
    if single(times) and multiple(indexes) then
        clause-indexes = preserve-same-clause(indexes, times[0])
        if exist(clause-indexes) then indexes = clause-indexes
        end if
    else if multiple(times) and single(indexes) then
        clause-times = preserve-same-clause(times, indexes[0])
        if exist(clause-times) then times = clause-times
        end if
    end if
    return ruleA(times, indexes)
end function
```

References

1. Ernst P, Siu A, Weikum G. Highlife: Higher-arity fact harvesting. In: Proceedings of the 2018 World Wide Web Conference. 2018, 1013–1022
2. Hassan N, Arslan F, Li C, Tremayne M. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017, 1803–1812
3. Mintz M, Bills S, Snow R, Dan J. 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. August 2009, 1003–1011
4. Aggarwal C C, Zhai C. Mining text data. Springer Science & Business Media, 2012
5. Miwa M, Bansal M. 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). August 2016, 1105–1116
6. Xu Y, Mou L, Li G, Chen Y, Peng H, Jin Z. Classifying relations via long short term memory networks along shortest dependency paths. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. September 2015, 1785–1794
7. Zhou P, Shi W, Tian J, Qi Z, Li B, Hao H, Xu B. Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). August 2016, 207–212
8. Zhang Y, Qi P, Manning C D. Graph convolution over pruned dependency trees improves relation extraction. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. October-November 2018, 2205–2215
9. Katiyar A, Cardie C. 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). July 2017, 917–928
10. Christopoulou F, Miwa M, Ananiadou S. A walk-based model on entity graphs for relation extraction. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). July 2018, 81–88
11. Zeng W, Lin Y, Liu Z, Sun M. Incorporating relation paths in neural relation extraction. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. September 2017, 1768–1777
12. Suchanek F M, Kasneci G, Weikum G. Yago: A large ontology from wikipedia and wordnet. Journal of Web Semantics, 2008, 6(3): 203 – 217. World Wide Web Conference 2007Semantic Web Track
13. Zhou D, Zhong D, He Y. Biomedical relation extraction: from binary to complex. Computational and mathematical methods in medicine,

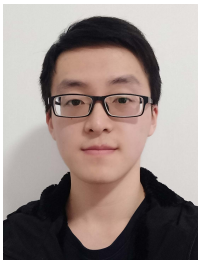
2014

14. McDonald R, Pereira F, Kulick S, Winters S, Jin Y, White P. Simple algorithms for complex relation extraction with applications to biomedical IE. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). June 2005, 491–498
15. Li J, Sun Y, Johnson R J, Sciaky D, Wei C H, Leaman R, Davis A P, Mattingly C J, Wiegers T C, Lu Z. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. Database the Journal of Biological Databases & Curation, 2016, 2016: baw068
16. Peng Y, Wei C H, Lu Z. Improving chemical disease relation extraction with rich features and weakly labeled data. Journal of Cheminformatics, 2016, 8(1): 53
17. Verga P, Strubell E, McCallum A. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). June 2018, 872–884
18. Cui L, Wei F, Zhou M. Neural open information extraction, 2018
19. RESHADAT V, HOORALI M, FAILI H. A hybrid method for open information extraction based on shallow and deep linguistic analysis. Interdisciplinary Information Sciences, 2016, 22(1): 87–100
20. Reshadat V, Faili H. A new open information extraction system using sentence difficulty estimation. COMPUTING AND INFORMATICS, 2019, 38(4)
21. Sun M, Li X, Wang X, Fan M, Feng Y, Li P. Logician: A unified end-to-end neural approach for open-domain information extraction. In: Proceedings of the Eleventh ACM International Conference on Web Search & Data Mining. 2018
22. Chen Y, Xu L, Liu K, Zeng D, Zhao J. Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). July 2015, 167–176
23. Blunsom P, Freitas d N, Grefenstette E, Hermann K M. A deep architecture for semantic parsing. In: Proceedings of the ACL 2014 Workshop on Semantic Parsing. 2014
24. Liang C, Berant J, Le Q, Forbus K D, Lao N. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). July 2017, 23–33
25. Wang Y, Berant J, Liang P. Building a semantic parser overnight. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). July 2015, 1332–1342
26. Xiao C, Dymetman M, Gardent C. Sequence-based structured prediction for semantic parsing. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). August 2016, 1341–1350
27. Berant J, Chou A, Frostig R, Liang P. Semantic parsing on Freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. October 2013, 1533–1544
28. Hershcovich D, Abend O, Rappoport A. A transition-based directed acyclic graph parser for UCCA. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). July 2017, 1127–1138
29. Zhu X, Sobihani P, Guo H. Long short-term memory over recursive structures. In: Bach F, Blei D, eds, Proceedings of the 32nd International Conference on Machine Learning. 07–09 Jul 2015, 1604–1612
30. Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). July 2015, 1556–1566
31. Agerri R, Rigau G. Robust multilingual named entity recognition with shallow semi-supervised features. Artificial Intelligence, 2016, 238: 63–82
32. Aguilar J, Beller C, McNamee P, Van Durme B, Strassel S, Song Z, Ellis J. A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards. In: Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation. 2014, 45–53
33. Girju R, Nakov P, Nastase V, Szpakowicz S, Turney P, Yuret D. Semeval-2007 task 04: Classification of semantic relations between nominals. In: Proceedings of the 4th International Workshop on Semantic Evaluations. 2007, 13–18
34. Hendrickx I, Kim S N, Kozareva Z, Nakov P, Ó Séaghdha D, Padó S, Pennacchiotti M, Romano L, Szpakowicz S. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions. 2009, 94–99
35. Wang Y, Liu X, Shi S. Deep neural solver for math word problems. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. September 2017, 845–854
36. Wang L, Zhang D, Zhang J, Xu X, Gao L, Dai B T, Shen H T. Template-based math word problem solvers with recursive neural networks. In: The Thirty-Third AAAI Conference on Artificial Intelligence. 2019, 7144–7151
37. Zeng D, Liu K, Chen Y, Zhao J. Distant supervision for relation extraction via piecewise convolutional neural networks. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. September 2015, 1753–1762
38. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997
39. Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. September 2015, 1412–1421
40. Cao Y, Li H, Luo P, Yao J. Towards automatic numerical cross-checking: Extracting formulas from text. In: Proceedings of the 2018 World Wide Web Conference, WWW '18. 2018, 1795–1804
41. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N,

Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17. 2017, 6000–6010
42. Zeiler M D. Adadelta: An adaptive learning rate method, 2012
43. Huang D, Yao J, Lin C, Zhou Q, Yin J. Using intermediate representations to solve math word problems. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. 2018, 419–428
44. Gu J, Lu Z, Li H, Li V O. Incorporating copying mechanism in sequence-to-sequence learning. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). August 2016, 1631–1640
45. Klein G, Kim Y, Deng Y, Senellart J, Rush A. OpenNMT: Open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations. July 2017, 67–72
46. Zheng S, Wang F, Bao H, Hao Y, Zhou P, Xu B. 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). July 2017, 1227–1236
47. Cao Y, Chen D, Li H, Luo P. Nested relation extraction with iterative neural network. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM). 2019

Yixuan Cao received the BE degree in transportation engineering from Tongji University in 2015 and now is a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include natural language processing and information extraction.

Dian Chen received the BE degree in IoT Engineering from ChongQing University in 2016 and now is a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests focus on Natural Language Processing, Deep Learning and Data Mining.



Zhengqi Xu received the BE degree in remote sensing from Beihang University, China in 2019 and now is an M-S student at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests focus on machine learning, information retrieval and information extraction.

Hongwei Li received the BE degree in software engineering from Fuzhou University, China in 2015 and now is a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests focus on machine learning, natural language processing and information extraction.

Ping Luo received the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences. He is an associate professor in the Institute of Computing Technology, Chinese Academy of Science (CAS). His general area of research is knowledge discovery and machine learning.