World Wide Web (2022) 25:195–218
https://doi.org/10.1007/s11280-021-00982-4

# An efficient and effective approach for multi-fact extraction from text corpus

Jianfeng Qu[1] · Wen Hua[2] · Dantong Ouyang[3] · Xiaofang Zhou[4]

Received: 10 May 2021 / Revised: 24 September 2021 / Accepted: 22 November 2021 /
Published online: 14 December 2021

## Abstract
Relation extraction (RE) is a fundamental task with various real-world applications. Although significant progress has been achieved in this research field, it is still limited to single-fact extraction. In practice, however, people tend to describe multiple relations in a single sentence. Apparently, multi-fact extraction is more reasonable yet challenging due to the mixture of diverse information. To address this issue, we introduce a novel syntax-based model for multi-fact extraction. Specifically, we propose a relational-expressiveness-based pruning strategy to refine the dependency parsing tree of each sentence, and then incorporate the customized and simplified syntax information into sentence encoding via Graph Convolutional Networks. Besides, distance embeddings are developed in our model to inform the extractor of the status of each word regarding different entity pairs in a sentence based on its shortest dependency path to the entities of interest. In addition, we explore fine-grained pooling strategy to integrate various evidences for the relation extractor to make accurate predictions. We conduct extensive experiments on the publicly-available datasets, and the experimental results verify the superiority of our model for multi-fact extraction in terms of both effectiveness and efficiency.

**Keywords** Relation extraction · Multi-fact · Pruning strategy · Dependency parse tree · Graph convolutional networks

✉ Dantong Ouyang
ouyd@jlu.edu.cn

1 School of Computer Science and Technology, Soochow University, Suzhou 215006, China

2 School of Information Technology and Electrical Engineering, The University of Queensland, St Lucia, QLD 4072, Australia

3 College of Computer Science and Technology, Jilin University, Changchun 130012, China

4 Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hongkong, China

## 1 Introduction

Relation extraction focuses on identifying relational facts for entity pairs of interest from plain texts. Since it can facilitate various downstream applications such as question answering (QA) [8, 12] and knowledge graph construction (KGC) [6, 29], the task has become a hot research topic in the field of natural language processing (NLP).

Previous works on relation extraction [13, 20] are mainly based on a simple assumption: each sentence expresses at most one relational fact, which, however, is inconsistent with the habit of human expression. In other words, people tend to simultaneously describe multiple relational facts in a single sentence for convenience. Consider the following sentence as an example.

***Example 1*** His father John, who was born in County Kerry, was educated at Saint Columbas College and his mother Mary, who was born in Courtmacsherry, County Cork, studied at the Moscow Conservatory.

An ideal relation extractor should be able to identify the following triplets: *born_in(John, County Kerry)*, *born_in(Mary, Courtmacsherry)*, *educated_at(John, Saint Columbas College)* and *educated_at (Mary, Moscow Conservatory)*. Although it seems quite easy to discover these triplets by human work, it brings great challenges to an automatic extraction model since multiple relational information is blended in one sentence. More specifically, the challenges lie in two aspects:

**Challenge 1**: How to effectively encode the sentence to equip the relation extractor with the ability of capturing links between target entity pair. In Example 1, when predicting the facts *educated_ at(John, Saint Columbas College)* and *educated_at (Mary, Moscow Conservatory)*, existing sequential encodings (e.g., convolutional neural networks (CNNs) and recurrent neural networks (RNNs)) pay more attention to features that are sequentially closer to the target entities, making the model deficient and inevitably produce misleading features for the extractor. Specifically, “Saint Columbas College” is closer to its irrelevant word “Mary” but farther away from its related word “John”. Additionally, “Moscow Conservatory” is far from any other entity of interest. In this case, the encoder tends to generate features which imply that “Mary” and “Saint Columbas College” have a certain relationship while any remaining entity pairs are unrelated. Moreover, the key words (i.e., “educated” and “studied”) for predicting the relation type “educated_in” also have long sequence distance to their head entities (i.e., “John” and “Mary” respectively). Obviously, sequential encoding methods cannot deal with the heterogeneous structure of multi-fact sentences.

To address this problem, we resort to the syntax of sentences and incorporate a tailored syntactic structure into sentence encodings through graph convolutional networks (GCNs). Specifically, we first obtain the dependency parse tree of a sentence using an external syntactic parser, as shown in Figure 1. Since the original intention of dependency tree is to represent the complex semantic structure of a sentence [7] rather than serving for the task of relation extraction, let alone multi-fact extraction, the obtained tree inevitably contains a large number of trivial and even confusing edges (e.g., $born \xrightarrow{auxpass} was$), resulting in excessive time and space consumption of GCNs, and even degrading the effectiveness of GCNs. Thus, conventional usage of the entire dependency tree structure without any pruning often brings too much useless information [18] while a straight treatment of only using short dependency path might ignore some crucial information [30]. Unlike them, we

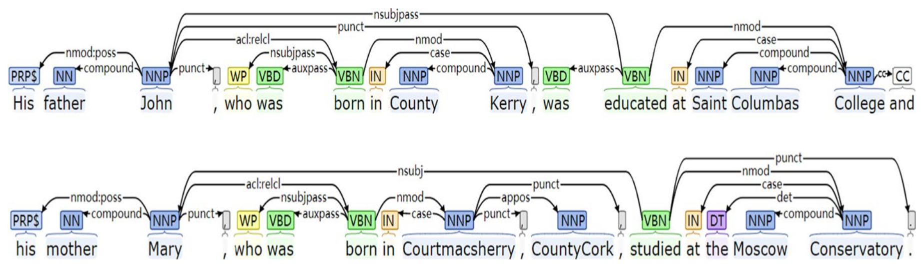

nsubjpass
punct
acl:relcl
nsubjpass
nmod
case
nmod
case
compound
nmod:poss
compound
punct
auxpass
compound
auxpass
compound
cc
PRP$ NN NNP , WP VBD VBN IN NNP NNP , VBD VBN IN NNP NNP NNP CC
His father John , who was born in County Kerry , was educated at Saint Columbas College and
nsubj
punct
nmod
case
det
compound
acl:relcl
nsubjpass
nmod
case
punct
appos
nmod:poss
compound
punct
auxpass
punct
PRP$ NN NNP , WP VBD VBN IN NNP , NNP , VBN IN DT NNP NNP .
his mother Mary , who was born in Courtmacsherry , CountyCork , studied at the Moscow Conservatory .

**Fig. 1** A dependency tree for the sentence "His father *John*, who was born in *County Kerry*, was educated at *Saint Columbas College* and his mother *Mary*, who was born in *Courtmacsherry*, County Cork, studied at the *Moscow Conservatory*"

propose a relational-expressiveness-based pruning strategy to refine the acquired tree and generate a tailored tree structure for multi-fact extraction. Based on this, for each word in the sentence, we combine all the relevant information (i.e., words that have syntactic arcs connected with the target word) to construct an informative vector representation. Such a representation achieves a deep view of which word is indeed semantically related to the target word instead of a shallow view of its immediate neighbors in the sentence. Considering the predicate "educated" in Example 1, our model can successfully identify the subject "John" and the adverbial "Saint Columbas College" via the respective connection arcs (e.g., *educated* $\xrightarrow{nsubjpass}$ *John*, *educated* $\xrightarrow{nmod}$ *Saint Columbas College*), and fuse them into the vector representation of "educated". In addition, our fusion process attempts to alleviate errors introduced by the external syntactic parser through an attention mechanism so as to assign less attention weight for erroneous links. Consequently, our GCNs can get rid of the limitation of sequence distance and efficiently encode these syntactic-related words more tightly.

**Challenge 2**: How to separately encode the same sentence regarding different target entity pairs. Since multiple information is mixed together in a single sentence, it is indispensable to produce corresponding sentence features for each entity pair of interest. However, previous methods utilize the final hidden state of RNNs or the global max pooling layer subsequent to CNNs as sentence features to serve as the evidence for predicting different target entity pairs. In other words, all the pairs are given the same information hybrid to identify the relationship between each pair, which confuses the extractor and makes the extraction less effective. To generate differentiated encodings for each entity pair, [23] simply uses shallow marker embeddings, which indicates whether the current word is the target entity or a common word, to obtain a customized feature representation. This kind of treatment is evidently insufficient to generate an effective sentence encoding since it regards all the words, except target entities, as equal. An alternative way is position embeddings proposed by [31], which considers the relative sequence distance to the target entities in the sentence. For instance, the relative distances from "educated" to the head entity "John" and the tail entity "Saint Columbas College" are 9 and 2, respectively. Obviously, such embeddings cannot always provide useful information, especially in the multi-fact situation, since the word far away from the target entity pair is not necessarily insignificant in expressing relational facts. In this case, position embeddings will inversely mislead the relation extractor.

To resolve this problem, we first develop distance embeddings to better reveal the importance of each word regarding different target entity pairs. In particular, we investigate the shortest path distances from the current word to the head and tail entities in the dependency tree, and integrate them into word embeddings as the initial input for the neural networks. For example, the shortest distances from “educated” to “John” and “Saint Columbas College” are equal to 1 (likely, the shortest distance from “studied” to “Mary” and “Moscow Conservatory” are also equal to 1), which provides neural networks with more accurate information about the importance of this word, compared to marker embeddings and position embeddings. Furthermore, we abandon the convention of using the final state or the global max pooling layer, and intend to construct distinct features for each target entity pair separately. To this end, words in a sentence will be grouped into different sets according to the relative position of the current entity pair. In fact, the components of each sets will be dynamically changed as we need to predict the relationship between various candidate entity pairs in a sentence. Then a fine-grained pooling strategy is developed to obtain the final sentence features for these candidate pairs. In this way, our pooling layer retains more valid information, and produces an individual sentence encoding for each pair of target entities in the sentence.

Our major contributions can be summarized as follows:

- We incorporate the syntactic information into multi-fact sentence encoding through GCNs, providing an effective sentence feature for relation extraction.
- We design a relational-expressiveness-based pruning strategy to generate a tailored dependency tree for GCNs, promoting both the effectiveness and efficiency of the entire model.
- We propose various techniques to handle the co-existence of multiple facts in a sentence. In particular, we introduce the distance embeddings to explicitly inform neural networks about the importance of each word regarding different target entity pairs, and investigate a fine-grained pooling strategy to separately consider every component in describing the desired relational fact.
- We conduct extensive experiments on the widely-used dataset, and the experimental results verify the superiority of our model compared with various strong baselines.

The remaining of the paper is organized as follows: we will elaborate on the details of our proposed model in Section 2. Our experimental results will be reported in Section 3, followed by a literature review in Section 4 and a brief conclusion of the work in Section 5.

## 2 Methodology

Suppose that a sentence $S$ is represented as a sequence of words: $\{w_1, w_2, w_3, \cdots, w_n\}$, and several entities are recognized from the sentence: $\{e_1, e_2, e_3, \cdots, e_m\}$. Considering the co-existence of multiple relational facts, the trained extractor should be able to identify the relation between each candidate entity pair (i.e., $r_1(e_1, e_2)$, $r_2(e_3, e_4)$, $\cdots$), where each $r$ represents either a predefined relation label (i.e., $r \in \{R\}$) or $NA$ (i.e., no relation between entities). To this end, we design a distinct and effective feature representation for each entity pair given the input sentence $S$. Figure 2 illustrates the framework of our proposed model. We will elaborate on each component of the model in the following.

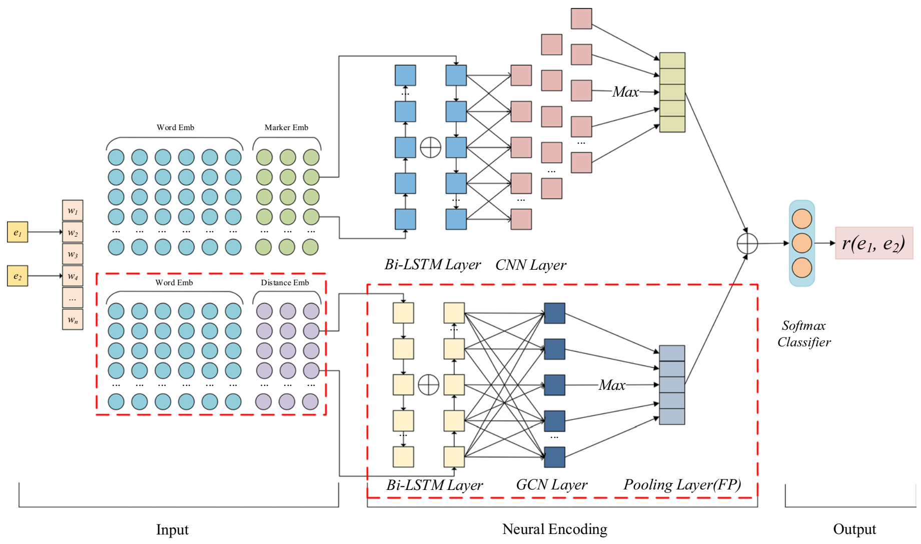


**Fig. 2** Framework of the proposed model. The red parts are the main contributions of our proposed model including distance embeddings, GCN layer on pruned dependency parse tree and fine-grained pooling strategy

## 2.1 Input representation

for each word in *s*, we transform it into a low-dimensional real-valued vector to serve as the input of the neural networks. specifically, the vector is composed of three major parts: word embeddings, marker embeddings and distance embeddings.

### 2.1.1 Word embeddings

Word embeddings employ distributed representations to convey the semantic information hidden in words. In general, the embeddings are learned from millions of unlabeled documents according to the co-occurrence statistics of words. In this work, we directly adopt a pre-trained tool, GloVe,[1] to obtain such representations. The sentence $S$ is represented as: $\boldsymbol{w}_1^E, \boldsymbol{w}_2^E, \boldsymbol{w}_3^E, \cdots, \boldsymbol{w}_n^E$, in which $\boldsymbol{w}_i^E \in \mathbb{R}^{de}$ and $de$ is the dimension of the vector $\boldsymbol{w}_i^E$.

### 2.1.2 Marker embeddings

Marker embeddings, proposed by [23], is utilized to denote whether a word is the head entity $e_1$, the tail entity $e_2$, or neither of them. The marker embeddings for word $w_i$ is represented as $\boldsymbol{w}_i^M \in \mathbb{R}^{dm}$ where $dm$ is the dimension of the vector $\boldsymbol{w}_i^M$. We randomly initialize the marker embedding matrix $\boldsymbol{M} \in \mathbb{R}^{3\times dm}$ (corresponding to three marker types: the head entity, the tail entity, and common words), and regard them as common parameters which will be updated during the training of the entire model.

[1] https://nlp.stanford.edu/projects/glove/

### 2.1.3 Distance embeddings

In practice, marker embeddings only point out the position of the entity pair of interest, while marking the remaining words (i.e., words not associated with the entity pair) with identical flags. Obviously, such a representation cannot sufficiently deliver crucial clues hidden in common words. A straightforward alternative is position embeddings, which calculates the relative distance between a word and the target entity pair in the sentence. However, position embeddings are sometimes problematic in relation extraction, especially under the circumstance of multi-fact. For instance, "born" and "educated" in Example 1 are of nearly the same importance for triplets *born_in(John, County Kerry)* and *educated_in(John, Saint Columbas College)*. Whereas, the distances between these two words and the head entity (i.e., *John*) are 2 and 9 respectively, resulting in totally different position embeddings for them. This phenomenon violates the intuition of position embeddings that the word closer to the entity pair plays a more important role in expressing their relation. To resolve this problem, we propose a new embedding method, namely distance embeddings, to better reflect the importance of each word in the sentence.

Given a sentence $S$, we first acquire its syntactic information, as shown in Figure 1, using an external dependency parsing tool (e.g., Stanford CoreNLP toolkit[2]). An adjacency matrix $\boldsymbol{A}$ is used to store the syntactic dependencies, where each word in $S$ is represented by a node and each arc between words is formalized as an edge. For simplicity, arc directions are ignored. We then compute shortest path distance from the current word $w_i$ to the head entity $e_1$ and the tail entity $e_2$, respectively, by conducting a Breadth-First-Search (BFS) on $\boldsymbol{A}$. We can see from Figure 1 that both "born" and "educated" have the same path distance (i.e., distance = 1) to the head entity (i.e., *John*) and tail entities (i.e., *County kerry* and *Saint Columbas College* respectively), which provides a more reasonable and informative position representation for extracting both relations. We denote the distance embeddings for each word as $\boldsymbol{w}_i^{\boldsymbol{P}} \in \mathbb{R}^{dp}$. The value of distance belongs to [0, max_sentence_length]. Then we randomly initialize a $dp$-dimensional vector for each value and construct a distance matrix $\boldsymbol{D}$. In this way, distance embeddings of each word in a sentence can be obtained by looking up $\boldsymbol{D}$. During training process, this embedding will be treated as normal parameters and updated via back-propagation on the objective function.

Based on these three types of embeddings, we generate the following compounds: $\boldsymbol{w}_i$ =$[\boldsymbol{w}_i^{\boldsymbol{E}}; \boldsymbol{w}_i^{\boldsymbol{M}}]$ and $\boldsymbol{w}_i'$=$[\boldsymbol{w}_i^{\boldsymbol{E}}; \boldsymbol{w}_i^{\boldsymbol{P}}]$, where [;] represents the concatenation operation. By regarding these compounds as inputs to the neural networks, our model can provide semantic message (word embeddings), together with sequence message (marker embeddings) and syntactic-dependency message (distance embeddings), respectively, for multi-fact extraction.

## 2.2 Dependency parse tree pruning

After processing sentences by an external dependency parsing tool, we can obtain a valuable dependency tree for each sentence, which is responsible for providing informative syntactic signals and beneficial for producing an effective sentence encoding. However, the original intention of the dependency tree is to represent the semantic structure of a sentence rather than serving for the task of relation extraction. As a consequence, the number

[2] https://github.com/stanfordnlp/CoreNLP

**Table 1** Statistics of dependency information

| | Train | Validation | Test |
|---|---|---|---|
| Sentences | 372,877 | 124,074 | 361,420 |
| Dependency edges | 9,245,458 | 3,076,170 | 9,035,076 |
| Average edges (per sentence) | 24.79 | 24.79 | 25.00 |

of dependency edges is extremely large in practice, resulting in massive space consumption and excessive computational cost when directly utilizing the tree for sentence encoding. Take the widely-used dataset of multi-fact sentences as an example.[3] We apply the Standford CoreNLP toolkit on the dataset and report the statistical results of the dependency information in Table 1. As we can see, there are 9,245,458 edges, 3,076,170 edges and 9,035,076 edges in the training, validation and testing set respectively. Furthermore, each edge connects to two endpoints (i.e., words), which means we need to consider twice of the syntactic dependency relationships (i.e., from parent node to child node of an edge, and vice versa) when encoding them into the sentence representation. Hence, it is indispensable to prune trivial or even misleading edges from the dependency tree while reserving salient information such that the tailored tree structure can better meet our demand for relation extraction.

Ideally, edges preserved in the dependency tree should be expressive of the relationships and distinctive of multiple facts. To this end, we design a heuristic relational-expressiveness-based (REB) pruning strategy for tree simplification based on the following intuitions:

– The nodes with higher degrees sit in the center of sentence semantics and play more crucial roles in describing a specific relation between an entity pair, and thus should be considered first in the pruning process (e.g., the nodes "born", "educated" and "studied" in Figure 1) and remove redundant or confusing edges connected to them.
– An edge is usually of less importance if it is connected to a leaf node (e.g., "$born \xrightarrow{auxpass} was$", "$Kerry \xrightarrow{case} in$", "$College \xrightarrow{case} at$" and "$John \xrightarrow{punct} ,$" in Figure 1), and thus is safe to be removed from the dependency tree. However, since the tree pruning should not break the semantic meaning of the original sentence, we still need to preserve edges that reflect semantics of entities and their relations. In particular, we keep edges in the following cases: (1) The syntax relationship on the edge is *compound* which is used for expressing noun-phrases; (2) The edge belongs to the shortest dependency path from the head entity to the tail entity; (3) The syntax relation is *advmod* reflecting an adverbial modifier of a word. This sometimes has a decisive role in predicting relations (e.g., the negative modifier "$born \xrightarrow{advmod} not$").
– If the edge's child node is a non-leaf node and meanwhile the nearest neighboring entity pairs (this is based on the tree structure regardless of the direction of edges) of the edge's endpoints are different, we need to delete the edge between these two nodes so as to avoid the confusing message-passing due to the existence of multiple facts. Consider the edge "$born \xrightarrow{conj:and} died$" in Figure 3. It is obvious that we should delete the edge between these two nodes (i.e., "born" and "died"). Otherwise, the semantic information of "born" and "died" will follow through this directly con-

[3] https://github.com/UKPLab/emnlp2017-relation-extraction

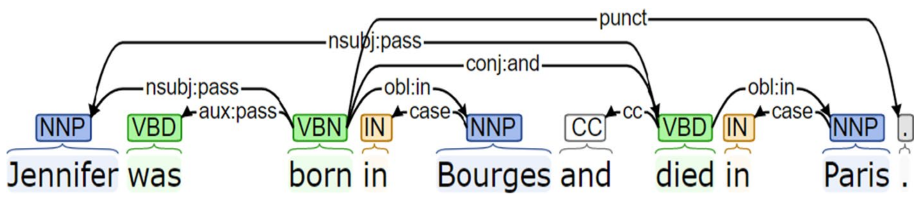


**Fig. 3** The original dependency tree for the sentence "*Jennifer* was born in *Bourges* and died in *Paris*"

nected edge and pose great challenges for the extractor to predict the facts: *born_in(Jennifer, Bourges)* and *die_in(Jennifer, Paris)*.

**Algorithm 1** REB tree pruning strategy.

```
Input: a dependency tree $\mathcal{T}$, candiate entity pairs $EP$: $\{ep_1, ep_2, \cdots, ep_n\}$
Output: a pruned dependency tree $\mathcal{T}'$.
1  $\mathcal{T}' \leftarrow \phi$
2  rank the nodes $N(\mathcal{T})$ in the descending order of their degree
3  add the nodes $N(\mathcal{T})$ into $\mathcal{T}'$
4  for each $u$ in $N(\mathcal{T})$ do
5      get the child nodes of $u$: $c(u)$
6      if $c(u) \neq \phi$ then
7          find the nearest neighboring entity pair of $u$ : $ep_i$ from $EP$ according to $c(u)$
8          for each $v$ in $c(u)$ do
9              $delete(u, v) \leftarrow 0$
10             get the child nodes of $v$: $c(v)$
11             if $c(v) = \phi$ and $(u, v) \notin$ special cases then
12                 $delete(u, v) \leftarrow 1$
13             else if $c(v) \neq \phi$ and $ep_i \neq \phi$ then
14                 find the nearest neighboring entity pair of $v$: $ep_j$ from $EP$ according to $c(v)$
15                 if $ep_j = \phi$ and $ep_i \neq ep_j$ then
16                     $delete(u, v) \leftarrow 1$
17             if $delete(u, v) \neq 1$ then
18                 add the edge $(u, v)$ into $\mathcal{T}'$
19 return $\mathcal{T}'$
```

Algorithm 1 describes the proposed REB pruning procedure. In this algorithm, lines 2-4 are based on the first intuition, lines 11-12 correspond to the second intuition, while lines 13-16 reflect the third intuition. Take the dependency tree in Figure 1 as an example. The number of initial dependency edges is 34. After performing the pruning algorithm on this tree, 20 edges are removed according to the aforementioned heuristics while only 14 salient edges are preserved to produce the simplified syntax structure for the follow-up sentence encoding. Another example is illustrated in Figure 4, which is a pruned tree structure using Algorithm 1 for Figure 3. In particular, we need to remove the following edges: "$born \xrightarrow{aux:pass} was$", "$Bourges \xrightarrow{case} in$", "$died \xrightarrow{cc} and$", "$Paris \xrightarrow{case} in$" and "$born \xrightarrow{punct} .$", w.r.t lines 11-12, and meanwhile ""$born \xrightarrow{conj:and} died$" w.r.t lines 13-16 in Algorithm 1.

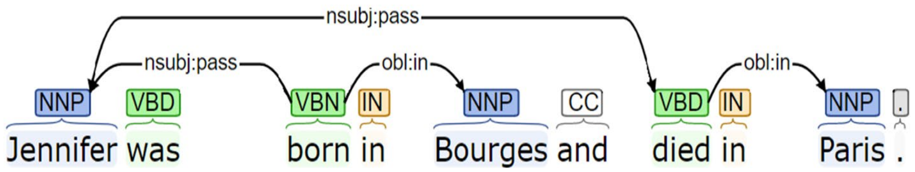

nsubj:pass
nsubj:pass
obl:in
obl:in
NNP VBD VBN IN NNP CC VBD IN NNP .
Jennifer was born in Bourges and died in Paris .

**Fig. 4** The pruned dependency tree for the sentence "*Jennifer* was born in *Bourges* and died in *Paris*" according to Algorithm 1

## 2.3 Neural networks

In order to obtain a more informative sentence feature for extracting multiple facts, we simultaneously encode sequence information (CNN layer) and syntactic information (GCN layer) from sentences in neural sentence encoding. In addition, considering the long-term dependency, we apply a bidirectional long short-term memory (Bi-LSTM) layer to the input of the entire model.

### 2.3.1 Bi-LSTM layer

LSTM, one of the typical recurrent neural networks, has been successfully applied to many NLP tasks, including machine translation, dialogue generation and information retrieval. We formalize LSTM as $LSTM(\boldsymbol{w_i}), (1 \leq i \leq n)$, and the hidden state of the $i$-th word as $h_i \in \mathbb{R}^{dh}$. In sequence expression, the future context is also important sometimes. Hence, we resort to Bidirectional LSTM: one for forward direction (i.e., $\rightarrow$) and another for backward direction (i.e., $\leftarrow$). The hidden states of these two directions for each word are concatenated as: $h_i=[\overrightarrow{h_i}; \overleftarrow{h_i}]$. We obtain two hidden states: $h_i$ and $h_i'$ corresponding to different inputs of Bi-LSTM: $\boldsymbol{w_i}$ and $\boldsymbol{w_i'}$.

### 2.3.2 CNN layer

As the key clues for relation extraction can appear anywhere within a sentence, we apply a CNN layer to extract every local features (namely consecutive information) with a sliding window $l$ over the sentence. Specifically, let vector $\boldsymbol{q_i}$ be the concatenation of a sequence of $l$-length hidden states:

$$\boldsymbol{q_i} = \boldsymbol{h}_{i-[l/2]:i+[l/2]} \quad (1 \leq i \leq n) \tag{1}$$

When the window slides near the boundary, we pad all-zero values for the out-of-range input $\boldsymbol{h}_i$ ($i \leq 1$ or $i \geq n$). A convolution operation is conducted over $\boldsymbol{q_i}$, and the output of CNN layer is $\{c^t_{w_1}, c^t_{w_2}, c^t_{w_3}, c^t_{w_4}, \cdots\}$, where the superscript $t$ means the $t$-th filter and the subscript $w_i$ is the central word of the convolutional window.

### 2.3.3 GCN layer

In traditional feature-based relation extraction, syntactic features are usually derived from a sentence using a dependency parser and then share the same importance as surface features when serving as the input for extractors. This proves that syntactic features are indeed beneficial for the task of relation extraction. However, most neural models only focus on surface sequential information while ignoring the syntax message due to the lack of an effective syntactic encoder. To this end, we utilize a GCN layer in our model to be applied on the graph structure of dependency parse tree.

We formalize the dependency parse tree of sentence $S$ as a graph $\mathcal{G}\Leftarrow\mathcal{V}\Leftrightarrow \mathcal{E}\Rightarrow$, where $\mathcal{V}$ and $\mathcal{E}$ denote the collection of nodes and edges respectively. Each node in $\mathcal{V}$ corresponds to a word in the sentence $\{w_1, w_2, w_3, \cdots, w_n\}$, and each edge $(w_i, w_j) \in \mathcal{E}$ represents the dependency path from the head node $w_i$ to the dependent node $w_j$ with the dependency function $df$ (i.e., $w_i \xrightarrow{df} w_j$). Considering the dependency path *educated*$\xrightarrow{nsubjpass+}$*John* in Figure 1, the head node is "educated", the dependent node is "John", and the dependency relation is "nsubjpass", which means "John" is the subject of "educated" in the sentence. In addition, we also consider the inversely directed edge $(w_j, w_i)$ (e.g., *John*$\xrightarrow{nsubjpass-}$*educated*), and the self-loop edge $(w_i, w_i)$ (e.g., *John*$\xrightarrow{loop}$ *John*) in this work. Then the final set of generated edges is denoted as follows:

$$E' = \mathcal{E} \cup E^{-} \cup Loop \tag{2}$$

where $\mathcal{E}^{-}$ is the set of inverse edges and *Loop* is the set of self-loop edges. Given that, the extended graph is represented as $\mathcal{G}\simeq\Leftarrow\mathcal{V}\Leftrightarrow\mathcal{E}\simeq\Rightarrow$. Let $u$, $v$ be two nodes in $\mathcal{V}$ , the $(k+1)$-th layer representation for $h_v$ in GCNs is computed by graph convolutional operation on the $k$-th layer:

$$h_v^{(k+1)} = \sum_{u \in \mathcal{N}(v)} \alpha_{(u,v)}^{(k)} \left( W_{l(u,v)}^{(k)} h_u^{(k)} + b_{l(u,v)}^{(k)} \right) \tag{3}$$

where $\mathcal{N}(v)$ is the neighbour set of $v$ in $\mathcal{G}\simeq$, $W_{l(u,v)}^{(k)}$ is the weight matrix, $b_{l(u,v)}^{(k)}$ is the bias vector, and $\alpha_{(u,v)}^{(k)}$ is the degree of relevance between $u$ and $v$. It is worth noting that, to avoid over-parameterization, the graph convolutional operation (i.e., $W_{l(u,v)}^{(k)}$ and $b_{l(u,v)}^{(k)}$) only varies in the direction of edges (i.e., head-to-dependent, dependent-to-head or self-loop) regardless of the syntactic function (e.g., *nsubjpass*, *auxpass*, *dobj*, $\cdots$) [4]. We formalize them as below:

$$W_{l(u,v)}^{(k)} = W_{dir(u,v)}^{(k)}, \qquad b_{l(u,v)}^{(k)} = b_{dir(u,v)}^{(k)} \tag{4}$$

We further define $\alpha_{(u,v)}^{(k)}$ according to the semantic relatedness hidden in $\mathcal{N}(v)$ and $v$:

$$\alpha_{(u,v)}^{k} = \frac{\exp(g_u^k \, g_v^k)}{\sum_{u' \in \mathcal{N}(v)} \exp(g_{u'}^k \, g_v^k)} \tag{5}$$

where $g_w^k = W_{dir(w,v)}^{(k)} h_w^{(k)} + b_{dir(w,v)}^{(k)}$. In this way, GCNs are able to capture the syntactically related words instead of limited to the sequentially adjacent words. Since we employ an external dependency parser to obtain these syntactic relationships, there inevitably contain some erroneous syntactic edges. Therefore, we introduce an attention mechanism during training, which will assign less weight to these suspected connections and reduce the side effect of errors caused by the dependency parser.

In each GCN layer, a node can only receive information from its immediate neighbours in the graph $\mathcal{G}\simeq$. Hence, we deepen the layers of GCNs to obtain long-distance information. That is, a node in the $j$-th layer receives information from neighbours at most $j$ hops away. However, the blind pursuit of deeper layers will inevitably bring serious problem of information redundancy (i.e., information flowing through edges will grow exponentially) and make the networks less efficient. The trade-off solution is to conduct GCNs on the top of the Bi-LSTM layer, denoted as:

$$h_u^0 = h_i' \quad u \in \mathcal{V} \tag{6}$$

### 2.3.4 Fine-grained pooling strategy (FP)

Previous works often use the final hidden state (RNNs) or global max-pooling (CNNs) to obtain the output of sentence features and feed into the relation extractor. These methods are effective in single-fact extraction, but might be problematic when dealing with multi-fact due to the lack of distinct treatments regarding different entity pairs. To give the extractor more explicit and distinct sentence encoding for each entity pair, we extend the piecewise pooling into multi-fact level. In particular, each word will be classified into three segments according to its relative position to the target entities: left, middle and right. Each segment plays different roles in describing a relational fact. Since multiple facts simultaneously exist in a single sentence, the components of each segment will be dynamically changed corresponding to different locations of entity pairs. Additionally, the vector representations of target entities are of great importance in predicting the relation between them. This property has been widely adopted in the task of knowledge graph completion (KGC) [33], which predicts relations using entity information without any context evidence. Motivated by these works, we further partition a sentence into five segments: *left window*, *entity*$_1$, *middle window*, *entity*$_2$, *right window*. Given the output from GCNs: $\{h^K_{w_1}, h^K_{w_2}, \cdots, h^K_{e_1}, h^K_{w_i}, h^K_{w_{i+1}}, \cdots, h^K_{e_2}, h^K_{w_j}, h^K_{w_{j+1}}, \cdots, h^K_{w_n}\}$, we conduct max-pooling operation on each segment, and concatenate them as the final sentence features from GCNs $\boldsymbol{V}_s^{gcn}$:

$$\begin{aligned}\boldsymbol{V}_s^{gcn} = \Big[ & max\Big(h^K_{w_1}, h^K_{w_2}, \cdots\Big); max(h^K_{e_1}); max\Big(h^K_{w_i}, h^K_{w_{i+1}}, \cdots\Big); \\ & max\Big(h^K_{e_2}\Big); max\Big(h^K_{w_j}, h^K_{w_{j+1}}, \cdots, h^K_{w_n}\Big)\Big]\end{aligned} \tag{7}$$

The process of fine-grained pooling is illustrated in Figure 5. The final obtained sentence features from GCNs $\boldsymbol{V}_s^{gcn}$ consist of information from each segment (i.e.,left window, entity$_1$, middle window, entity$_2$, right window), which provide sufficient and individual clues for the extractor to make predictions.

Similarly, with the fine-grained pooling strategy for the output of CNN layer, we obtain the final sentence features from CNNs as $\boldsymbol{V}_s^{cnn}$. We then concatenate these two parts of features to serve as the input of the extractor:

$$\boldsymbol{V}_s = [\boldsymbol{V}_s^{gcn}; \boldsymbol{V}_s^{cnn}] \tag{8}$$

## 2.4 Objective function and optimization

Given a training bag $T$, we define the conditional probability $p(r|\langle e_1, e_2\rangle, \boldsymbol{V}_s; \theta)$ via a softmax operation to compute the confidence of each possible relation:

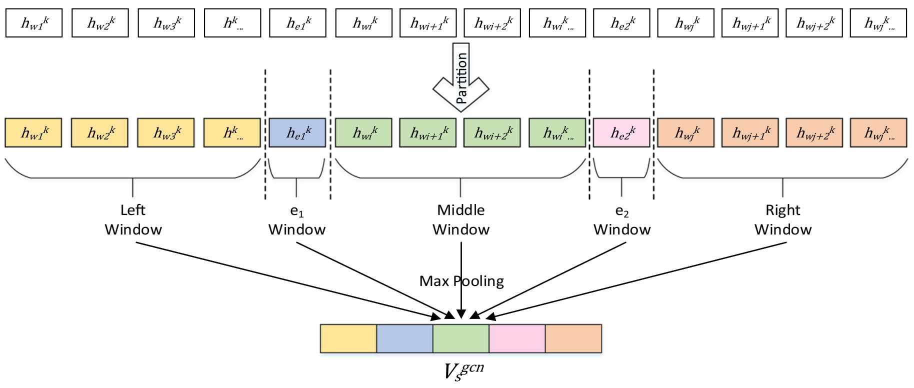


**Fig. 5** The process of fine-grained pooling. The process is subsequent to CNN layer and GCN layer, and the information flows from the top to the bottom. The output of pooling layer serves as input for the softmax layer

$$p(r|\langle e_1, e_2\rangle, \boldsymbol{V}_s;\theta) = \frac{\exp(o^r)}{\sum_{k=1}^{|R|}\exp(o^k)} \tag{9}$$

where $|R|$ is the total number of possible relations and $\boldsymbol{o} = \boldsymbol{Q} \cdot (\boldsymbol{V}_s \circ \boldsymbol{D}) + \boldsymbol{d}$. $\boldsymbol{Q}$ is a transform matrix from features to relations. Each value in $\boldsymbol{Q}$ represents the weight of the corresponding feature for predicting a specific relation label. $\boldsymbol{d}$ is a bias vector, and $\boldsymbol{D}$ is a dropout vector of Bernoulli random variables with probability $p$ used for regularization. $o^r$ represents the confidence that the sentence $S$ expresses the relation $r\langle e_1, e_2\rangle$.

Finally, we define the objective function for relation extraction using cross-entropy as follows:

$$J(\theta) = \sum_{i=1}^{|S|}\sum_{j=1}^{|R_i|} log\ p(r_{ij}|S_i;\theta) \tag{10}$$

where $r_{ij}$ is a possible relation label instantiated in $S_i$, $|S|$ denotes the total number of training sentences and $|R_i|$ is the number of relations in $S_i$. The entire model is trained by the Adam optimizer over mini-batches $B$ to maximize the objective function.

## 3 Experiments

We conduct extensive comparative experiments on a publicly available dataset to verify the effectiveness of the proposed method. In this section, we first introduce the dataset and the evaluation metrics used in the experiments. Then we describe some details about the model implementation, followed by our experimental results and discussions.

### 3.1 Dataset and evaluation metrics

Since our model is designed to resolve the multi-fact problem, we use a newly developed and challenging dataset Sorokin'17 proposed by [23], which mainly collects sentences containing multiple relational facts. The data is generated by aligning Wikidata and Wikipedia. For each sentence in an article, they extract link annotations from articles and retrieve Wikidata entity IDs corresponding to the linked articles. The association between Wikidata entities and Wikipedia articles is built by unambiguous one-to-one mapping. Additionally, for each pair of entities, they query Wikidata for relation types that connect them and discard an occurrence of an entity pair if the relation is ambiguous (i.e., an entity pair has multiple relation labels in Wikidata) [24]. Eventually, they randomly split the dataset into training, validation and testing sets. The statistics of this dataset is shown in Table 2. We can see that the phenomenon of sentences that express multiple relational facts is very common in the dataset. The number of relation types is 354 (out of approximately 1700 non-meta relation types in the Wikidata scheme), including a NA label (i.e., no relation between the two entities).

Following [15, 17, 23], we adopt held-out evaluation. In testing phase, the precision and recall are calculated by comparing the predictions with the relational facts in Wikidata. Additionally, each relation label shares the same proportion in the composition of precision and recall so that the results will not be influenced by uneven distribution of testing data. To sufficiently demonstrate the performance of each model, we evaluate them using two metrics: precision/recall curves, P@N metrics and F1 scores.

### 3.2 Implementation details

In this work, we straightforwardly use the pre-trained word embeddings, Glove, to obtain the distributed representation of each word in the dataset. The marker and distance embeddings are randomly initialized and updated as normal parameters during training. Additionally, to acquire the entire dependency tree of sentences, we employ the well-known syntactic parser: Stanford CoreNLP.

We tune all the hyper-parameters on the validation data. In detail, we use a grid search to determine the optimal parameters. The parameter settings are as follows: word dimension *de*: 50; marker dimension *dm*: 3; distance dimension *dp*: 3; convolutional filter: 256; the number of LSTM layer: 1; the number of GCN layer: 1; the number of CNN layer: 1; hidden state in LSTM layer: 128; dropout probability *p*: 0.5; learning rate: 0.01 and batch size *B*: 64.

### 3.3 Baselines

We compare our model, denoted as *SAE*, with several state-of-the-art relation extractors as listed below:

– *CNN-ATT* is a fine-grained CNN model [15] previously tested on single-fact extraction and achieves the state-of-the-art performance on the dataset [22].
– *Context-Sum* assumes that facts expressed in a sentence are relevant and combines all the encodings for the sentence by a simple addition [23].
– *Context-Aware* aims to capture the relevance between facts in a single sentence, which uses attention mechanism on sentence-level to obtain context-aware sentence representation for prediction [23].

**Table 2** Statistics of the experimental dataset

| | Train | Validation | Test |
|---|---|---|---|
| Sentences | 372,877 | 124,074 | 361,420 |
| Relation instances | 578,199 | 190,160 | 600,804 |
| Multi-fact sentences | 34% | 32% | 39% |

- *GP-GNN* is designed to inference multi-hop relations in a sentence via graph neural networks [34].
- *RECON* uses a graph neural network to learn representations of both the sentence as well as facts stored in a KG [5].

To guarantee a relatively fair comparison, we use their publicly released source codes[4] for these baselines.

## 3.4 Effectiveness of the proposed model

### 3.4.1 Comparison with baselines

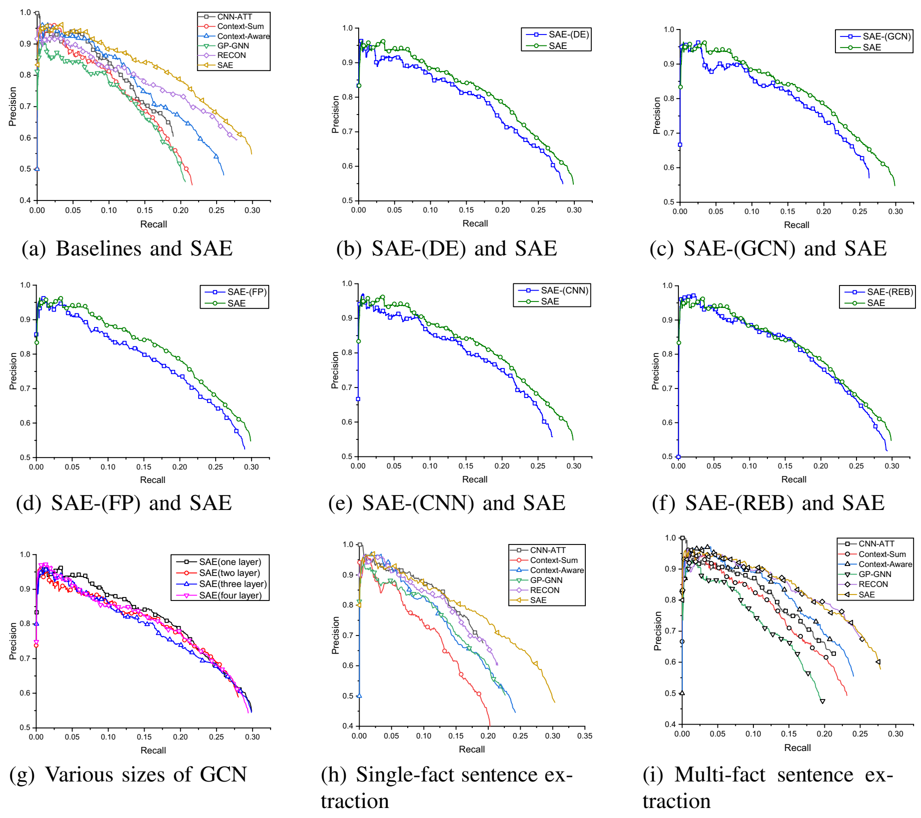
The precision/recall curves for *SAE* and all baselines are shown in Figure 6(a). We can see that: (1) *SAE* substantially outperforms all the baselines, demonstrating that our model is an effective way to capture syntactic information. In general, *SAE* obtains over 2% improvement on precision against *RECON* at the same recall. This is due to the fact that our model can construct a distinctive and informative feature representation for each entity pair in the same sentence, enabling the extractor to make precise predictions. (2) The recent method, i.e., *GP-GNN*, seems to be not comparable to other methods. This is because the method is effective only when the number of entities in the sentence is larger than 3 and there is at least one circle in the ground-truth label of the sentence. Apparently, the constraint is too strict and loses the ability of generalization. In contrast, *SAE* does not have any specific requirements on the dataset, which can be more fit to the real-world data.

### 3.4.2 Effectiveness of each component

To separately evaluate the effectiveness of each component in our model, we design the following variants of *SAE*: *SAE-X*, which means the removal of the component *X* and *X* can be *DE*, *GCN*, *FP*, *CNN* and *REB*.

We depict the precision/recall curves for *SAE* and its variants in Figure 6(b-f). We can see that: (1) the performance of *SAE-(DE)* is inferior to *SAE*. This is because marker embeddings, which only point out the positions of target entities and treat all the remaining words with the same marker, are too coarse for the networks to identify the key clues in sentences. In contrast, our distance embeddings consider the dependency path distance to the entities of interest, and are able to inform the networks of the importance of each word with respect to the entity pair. (2) *SAE-(GCN)* is constantly worse than *SAE*. This result demonstrates the necessity of syntactic features in multi-fact extraction. Besides,

[4] Available at https://github.com/UKPLab/emnlp2017-relation-extraction & https://github.com/thunlp/gp-gnn



(a) Baselines and SAE (b) SAE-(DE) and SAE (c) SAE-(GCN) and SAE (d) SAE-(FP) and SAE (e) SAE-(CNN) and SAE (f) SAE-(REB) and SAE (g) Various sizes of GCN (h) Single-fact sentence extraction (i) Multi-fact sentence extraction

**Fig. 6** Precision/recall curves of models

the GCN layer is an effective approach to incorporating syntax into neural networks. (3) The precision of *SAE-(FP)* is lower than *SAE*, especially when recall is low, which reveals the confidence and stability of *SAE-(FP)* is not enough. The reason is that global max-pooling strategy mixes all the information together regardless of the difference between entity pairs. Moreover, global max-pooling strategy loses too much informative clues even if GCNs can provide syntax information. (4) *SAE* outperforms *SAE-(CNN)* in all the region of the curves, which proves that every local part feature is of great importance in relation extraction, including multi-fact extraction, and provides useful sequential information for predicting relations. (5) *SAE* is superior than *SAE-(REB)* which straightforwardly adopts the initial entire dependency tree. This superiority indicates that our designed REB pruning strategy is able to filter these useless and even misleading edges and retain the informative dependency relationships which is beneficial to multi-fact extracting model.

To give a further inspection of the pruning strategy, we produce the number of syntactic edges belonging to each part of the dataset with or without pruning process in Figure 7. We can find that the number of syntactic edges dramatically decreases with our elaborated pruning strategy, approximating to 1/3 of the original. The result indicates that the trivial and confusing syntactic relationship is a severe problem regarding multi-fact extraction, and our pruning process effectively discerns them so as to produce a customized tree structure for the GCN layer. On the other hand, although the volume of syntactic information

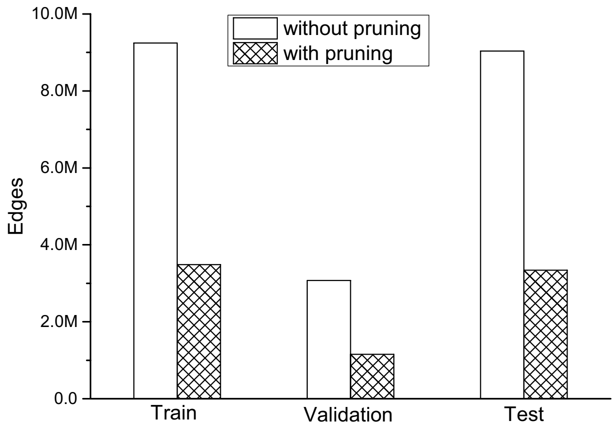


**Fig. 7** The number of syntactic edges before and after pruning

has been reduced, the performance of *SAE* is inversely improved (shown in Figure 6(f)) which sufficiently demonstrates the necessity of the pruning.

#### 3.4.3 Effectiveness of size of GCN layer

In addition, we evaluate *SAE* with different size of GCN layer. The results are shown in Figure 6(g). Obviously, the deepening of GCNs does not bring any improvement to the performance of the extractor, which is consistent with our anticipation that LSTM can help GCNs capture the long-term dependency information and improve both the efficiency (i.e., deeper GCN layers require more training time) and the effectiveness of GCNs. Besides, deeper GCN layers result in the redundancy of information and make the networks less effective.

#### 3.4.4 P@N scores

Following [15, 17], we also use P@N metrics to evaluate these models. In particular, the predictions of each model are ranked in descending order of confidence scores. Then we fetch the top N results and check the accuracy of these predictions, which is shown in Table 3. As expected, in P@200, P@300 and P@500, *SAE* always obtains higher precision than all the previous methods and the variants, and eventually achieves the highest average precision which is 3.4 points higher than *CNN-ATT*, 2.1 points higher than *Context-Aware*, 10.7 points higher than *GP-GNN* and 8.0 points higher than *RECON*.

#### 3.4.5 Highest F1 scores

Table 4 shows the highest F1 scores for the evaluated models. Among them, *SAE* achieves the highest F1 score (10.2% higher than *CNN-ATT*, 9.5% higher than *Context-Sum*, 4.9% higher than *Context-Aware*, 10.4% higher than *GP-GNN* and 1.1% higher than *RECON*). We conclude that our model *SAE* improves the performance of the extractor and make up

**Table 3** P@N scores for relation extraction

| P@N(%) | 200 | 300 | 500 | Average |
|---|---|---|---|---|
| *CNN-ATT* | 93.5 | 90.8 | 78.4 | 87.6 |
| *Context-Sum* | 92.1 | 85.7 | 73.6 | 83.8 |
| *Context-Aware* | 92.8 | 92.2 | 81.7 | 88.9 |
| *GP-GNN* | 84.0 | 83.3 | 73.6 | 80.3 |
| *RECON* | 89.2 | 83.4 | 76.3 | 83.0 |
| *SAE-(DE)* | 91.8 | 87.1 | 79.3 | 86.1 |
| *SAE-(GCN)* | 87.9 | 88.2 | 75.9 | 84.0 |
| *SAE-(FP)* | 91.1 | 85.7 | 75.6 | 84.1 |
| *SAE-(CNN)* | 90.8 | 86.9 | 78.3 | 85.3 |
| *SAE-(REB)* | 93.6 | 90.0 | 85.1 | 89.6 |
| *SAE* | 94.0 | 93.7 | 85.2 | 91.0 |

for the disadvantages of previous shallow sentence encodings. Additionally, each of proposed components (e.g, distance embeddings, GCNs, fine-grained pooling and REB pruning) can facilitate the task of relation extraction, and the combination is best.

### 3.4.6 Case study

To further verify the effect of the proposed model, we conduct a deeper check on the predictions. Table 5 lists some examples of the results produced by *Context-Aware* and *SAE* respectively: (1) In *S1*, *Context-Aware* only retrieves the last expressed fact (i.e., *place_of_bir-th(Nathan Bedford Forrest, Chapel Hill)*) while missing the prior fact (i.e., *member_of(Nathan Bedford Forrest, Ku Klux Klan)*). This is because *Context-Aware* uses the final state of LSTM as the output of the sentence feature which is too coarse and loses prior useful clues. Unlike them, *SAE* crafts a fine-grained pooling strategy and is able to construct an individual encoding for each pair of entities. (2) *S2* poses a great challenge for *Context-Aware* as the opposite facts simultaneously existed in the sentence are easy to be mistakenly labelled without syntax. Consequently, *Context-Aware* produces an incorrect prediction: *place_of_birth(Geoffroy Tory, Paris)*. In contrast, *SAE* is of enough capability to recognize that “Bourges” is the modifier for “born” while “Paris” is the modifier for “died”(as shown in Figure 8), and achieves the desired results. Through GCNs, the vector representation of the entity “Bourges” contains the semantic clues of “born” and “Paris” includes the semantic information of “died”. Further, the fine-grained pooling strategy retains the features of target entities. Finally, our model successfully predicts true relationships between these two pairs of interest without any hesitation. (3) Similarly, *S3* also needs syntactic features together with distance embeddings to help with the process of sentence encoding. *Context-Aware* gives the wrong prediction of *date_of_death(62 AD, Nero)* due to the lack of syntax. (4) We apply only one GCN layer in our model, to avoid information redundancy and excessive parameterization (Figure 6(g)). Hence, it is less accurate when dealing with entity pairs that are too far away in the dependency parse tree, which actually requires a very deep GCN layer to transmit information. For example, in *S4*, the dependency path from the actress “Sienna Miller” to the movie “Casanova” is too long. Therefore, our model *SAE* cannot predict the relational fact: *cast_member(Casanova, Sienna Miller)*. But this phenomenon is infrequent in the dataset.

**Table 4** F1 scores for relation extraction

| Model | Precision(%) | Recall(%) | F1(%) |
|---|---|---|---|
| *CNN-ATT* | 61.8 | 18.9 | 28.9 |
| *Context-Sum* | 48.3 | 21.3 | 29.6 |
| *Context-Aware* | 53.6 | 25.1 | 34.2 |
| *GP-GNN* | 51.6 | 19.9 | 28.7 |
| *RECON* | 59.5 | 27.9 | 38.0 |
| *SAE-(DE)* | 58.1 | 28.0 | 37.8 |
| *SAE-(GCN)* | 59.6 | 26.2 | 36.4 |
| *SAE-(FP)* | 56.1 | 28.5 | 37.8 |
| *SAE-(CNN)* | 59.4 | 26.4 | 36.6 |
| *SAE-(REB)* | 55.0 | 29.0 | 38.0 |
| *SAE* | 60.1 | 29.0 | 39.1 |

### 3.4.7 Effectiveness on single-fact extraction and multi-fact extraction

Although our model is for multi-fact extraction, we believe that our model is also effective in single-fact extraction. To prove our claim, we divide the dataset into two parts according to the number of relational facts expressed in the sentence: single-fact sentences and multi-fact sentences. We separately test our model on these two datasets. The precision/recall curves for *SAE* and all the baselines on single-fact sentences and multi-fact sentences are shown in Figure 6(h) and (i), respectively.

In Figure 6(h), when the recall is low, the precision of *SAE* is comparable to *CNN-ATT* that achieves the previous state-of-the-art performance on single-fact extraction. When the recall is greater than 0.13, *SAE* achieves much higher precision than *CNN-ATT*. And eventually, the recall of *SAE* is over 10% greater than *CNN-ATT*. It indicates that syntax information and fine-grained pooling are useful for single-fact extraction. In particular, when the structure of a sentence is complex, syntax information can help the extractor to find the semantically related words with respect to the two entities, which is not limited by sequence distance, so that the extractor is able to discover more relational facts (higher recall). And fine-grained pooling strategy retains valid information, rather than arbitrarily discards many useful clues in the previous method. In contrast, *Context-Aware* is constantly and substantially inferior to *CNN-ATT*. It means that the scope of *Context-Aware* is limited to multi-fact extraction and the model is not a good option for single-fact extraction.

In Figure 6(i), as expected, *SAE* is superior to all the baselines, including *RECON*. Finally, our model obtains the highest recall score. It demonstrates that GCNs can effectively embed syntax information into sentence encoding, and the generated syntactic features is beneficial for the extractor to discern the key cues for predicting different relations expressed in one sentence.

According to the above experimental results, the proposed model *SAE* is of sufficient capacity to deal with single-fact extraction, multi-fact extraction and hybrid-fact extraction (the evidence is shown in Figure 6(a)), and achieves the new state-of-the-art performance.

## 3.5 Results on ACE2005

We also test our model on another widely-used dataset, namely ACE2005.[5] This dataset has 4525 training, 705 validation, 3720 testing sentences of which 932, 112 and 319

[5] https://catalog.ldc.upenn.edu/LDC2006T06

**Table 5** Examples of predictions from *Context-Aware* and *SAE*

| Sentences | Relational facts | *Context-Aware* | *SAE* |
|---|---|---|---|
| *S1*: It is named for Nathan Bedford Forrest, a Confederate general and first | *member_of(Nathan Bedford Forrest, Ku Klux Klan)* | | √ |
| Grand Wizard of the Ku Klux Klan, who was born in Chapel Hill. | *place_of_birth(Nathan Bedford Forrest, Chapel Hill)* | √ | √ |
| *S2*: Geoffroy Tory (also Geofroy, Latin Godofredus Torinus) was born in Bourges | *place_of_ birth(Geoffroy Tory, Bourges)* | √ | √ |
| around 1480 and died in Paris before 14 October 1533. | *place_of_death(Geoffroy Tory, Paris)* | | √ |
| *S3*: Roman Emperor Nero ordered the death of | *date_of_death(62 AD, Octavia)* | | √ |
| his first wife, Octavia, soon after divorcing her in 62 AD. | *position_held(Nero, Roman Emperor)* | √ | √ |
| *S4*: He also played a lard merchant named | *cast_member(Casanova, Sienna Miller)* | | |
| Papprizzio in Lasse Hallström [director]'s | | | |
| Casanova [movie], who competes with | *cast_member(Casanova, Heath Ledger)* | √ | √ |
| Casanova[character] (Heath Ledger[actor]) | | | |
| for marriage to Francesca[character] | *director(Casanova, Lasse Hallström)* | √ | √ |
| (Sienna Miller[actress]). | | | |

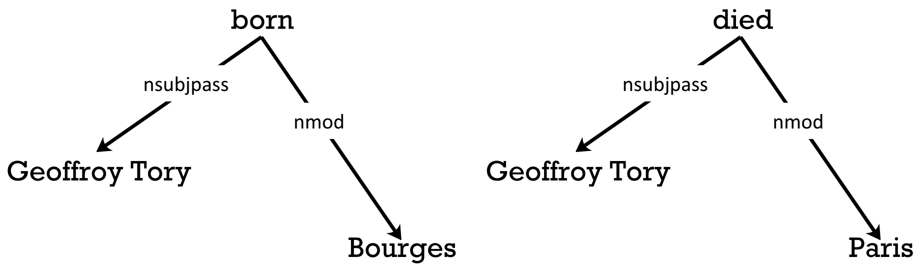


**Fig. 8** Partitial dependency tree of “Geoffroy Tory was born in Bourges around 1480 and died in Paris before 14 October 1533”

sentences describe multiple facts, respectively. Moreover, we use a new competitive baseline *BERT-SP* [26] which adopts a pre-trained language model BERT and completes this task with one-pass encoding. To achieve a relatively fair comparison, we replace our original Bi-LSTM with BERT. The experimental results are shown in Table 6. From this table, we can see that our model *SAE(BERT)* gets the highest F1 score that is 2.5 points higher than *BERT-SP*. We believe that the results further demonstrate the effectiveness of our proposed model.

### 3.6 Efficiency of the proposed model

To reveal the efficiency of the proposed model, we calculate the entire running time for a training iteration w.r.t an increasing size of GCN layer. For fairness, all the models are implemented by PyTorch and executed on a linux sever with a GeForce RTX 2080 Ti. The results are shown in Figure 9. We can see that with the size of GCN layer increases, the consumption of time in an training iteration will grow in an approximately straight line (around 1500s per GCN without pruning). We argue that the results demonstrate the necessity of our usage of the Bi-LSTM which can further help GCNs to capture long-term dependency so that the entire model achieves the best performance with only one GCN layer (shown in Figure 6(g)), avoiding the demand of deep GCNs and promoting the efficiency of the entire model. From the perspective of information redundancy, since our proposed REB pruning strategy dramatically reduces the trivial and misleading syntactic edges (shown in Figure 7), the training time has been greatly shortened compared with the original model. More specifically, the time difference between these two models is: 500s , 1312s, 1986s and 2427s for one, two, three and four GCN layers, respectively. In summary, with the help of Bi-LSTM and REB pruning strategy, our syntax encoder can be efficient in tackling with multi-fact extraction.

## 4 Related work

In this section, we provide a brief summarization of existing methods for relation extraction, including feature-based models, neural models, multi-fact extraction and open information extraction (OpenIE).

Feature-based relation extraction Relation extraction is one of the most important research tasks in NLP. Many efforts based on supervised learning have been invested to boost the performance of relation extractors [1, 14, 21]. These models have the ability to

**Table 6** Results on ACE2005

| **Model** | Precision(%) | Recall(%) | F1(%) |
|---|---|---|---|
| *CNN-ATT* | 60.0 | 59.4 | 59.7 |
| *Context-Sum* | 46.8 | 38.2 | 42.1 |
| *Context-Aware* | 48.5 | 25.3 | 33.3 |
| *GP-GNN* | 64.3 | 60.9 | 62.6 |
| *BERT-SP* | 68.9 | 66.8 | 67.8 |
| *SAE(BERT)* | 71.8 | 68.8 | 70.3 |

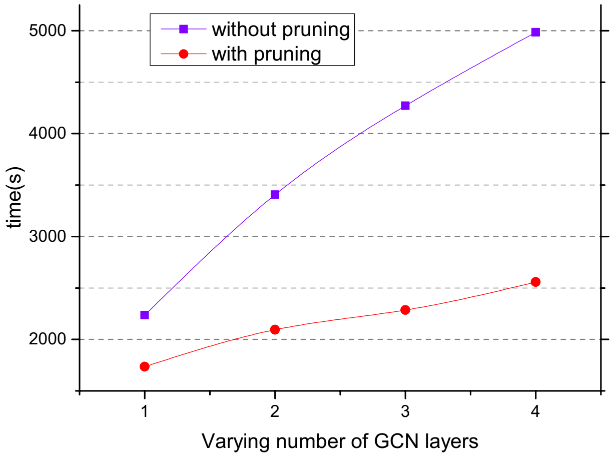


**Fig. 9** Time of an iteration w.r.t varying size of GCN layer

achieve outstanding performance on domain-specific data. However, the training data used in these models are labelled by human effort and cannot scale to big data. To address this issue, [17] develop distant supervision strategy which heuristically aligns an existing KB with free texts to automatically produce labelled data. For the aligned relation mentions, lexical and syntactic features are extracted by an external NLP toolkit and are vectorized using the bag-of-words representation. Later works focus on alleviating noise problem in distant supervision [24]. More recently, [20] considers feature sparsity in traditional feature representation.

Neural relation extraction Recently, neural networks have been successfully applied to many NLP tasks, including knowledge graph completion [9] and text classification [28]. To avoid hand-designed features, researchers have investigated the possibility of using neural networks to automatically learn features for relation extraction: RNNs [10], CNNs [15] and LSTM [18]. Some works have already noticed that syntactic features can also take effect in neural models and attempted to incorporate dependency tree structure into neural model [30]. However, their methods only focus on shortest dependency path (SDP), wasting too much information that is not included in SDP. More related works are [25, 32], which also adopt GCNs to encode syntax feature. The main differences between our work and the existing solutions are as follows: (1) These methods are based on single-fact extraction. In contrast, in this paper, we focus on a more promising and challenging topic: multi-fact extraction. (2) Except for GCNs, we develop REB pruning strategy, distance embeddings and dynamic fine-grained pooling strategy to construct a more effective and efficient sentence representation.

Multi-fact extraction Multi-fact extraction aims at recognizing relations of multiple pairs of entity mentions from text. Sorokin and Gurevych [23] construct a dataset of multi-relation per sentence and introduce a neural network architecture that considers other relations from the same context. After that, many works have been proposed to solve this task. Wang et al. [26] complete this task with only one-pass encoding on top of BERT. Due to the success of GNN, [5] study the effect of KG context using Graph Neural Networks(GNN) to learn representations of both the sentence as well as facts stored in a KG. Since not all KG context forms are necessary for every input sentence, KGPool [19] utilize a self-attention mechanism in a GCN for selecting a sub-graph from the KG to extend the sentential context. Others also perform named entity recognition and relation extraction collaboratively. Liu et al. [16] use GCNs to integrate entity information and multiple pairs of relations information to enhance the performance of both. Unlike their usage of GNN for encoding the structure of external KG, here, we exploit the dependency information from sentences and craft GCN to effectively obtain this crucial information.

Open information extraction OpenIE is a well-known technology in relation extraction, which is often regarded as unsupervised strategy and does not need any training data [3, 11, 27]. Among them, [2] is one of the most relevant to our work. More specifically, their model is able to extract multiple relational facts in one single sentence. However, they use the surface text linking two entities in a sentence as the relation name, which results in a large quantity of useless results (e.g., “have”, “get” and “has” as the types of relations). Hence, the outputs cannot be easily mapped to a particular knowledge base. In contrast, our work focuses on distantly supervised relation extraction that is able to extract instances with reference to given relations and achieves high precision and recall in practice. Therefore, our model is a more suitable way to construct knowledge graph.

## 5 Conclusion

In this paper, we craft a relational-expressiveness-based pruning strategy to build a tailored syntactic tree structure, and then incorporate syntax information into the task of multi-fact extraction through GCNs. Besides, distance embeddings are developed to inform the networks of the status of each word in a sentence with respect to different target entity pairs. In addition, we investigate a fine-grained pooling strategy to produce a distinctive and informative sentence encoding for the extractor so that the extractor can make accurate predictions with explicit evidence. We conduct extensive experiments and the experimental results demonstrate that the efficiency and effectiveness of our proposed method.

## References

1. Agichtein, E., Gravano, L.: Qxtract: A building block for efficient information extraction from plain-text databases. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego, California, USA, June 9-12, 2003 (2003)
2. Angeli, G., Premkumar, M.J.J., Manning, C.D.: Leveraging linguistic structure for open domain information extraction. In: Proceedings of ACL, pp. 344–354 (2015)

3. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego, California, USA, June 9-12, 2003, pp. 337–348 (2003)
4. Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., Sima'an, K.: Graph convolutional encoders for syntax-aware neural machine translation. In:Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 1957–1967 (2017)
5. Bastos, A., Nadgeri, A., Singh, K., Mulang, I.O., Shekarpour, S., Hoffart, J., Kaul, M.: RECON: relation extraction using knowledge graph context in a graph neural network. In: Leskovec, J., Grobelnik, M., Najork, M., Tang, J., Zia, L. (eds.) WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021, ACM / IW3C2, pp. 1673–1685 (2021)
6. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008, pp. 1247–1250 (2008)
7. Chen D, Manning CD A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 740–750 (2014)
8. Cui, W., Xiao, Y., Wang, H., Song, Y., Hwang, S., Wang, W.: KBQA: learning question answering over QA corpora and knowledge bases. Proc VLDB Endow **10**(5), 565–576 (2017)
9. Ebisu, T., Ichise, R.: Generalized translation-based embedding of knowledge graph. IEEE Trans Knowl Data Eng **32**(5), 941–951 (2020)
10. Ebrahimi, J., Dou, D.: Chain based RNN for relation classification. In: Proceedings of NAACL, pp. 1244–1249 (2015)
11. Gulhane, P., Rastogi, R., Sengamedu, S.H., Tengli, A.: Exploiting content redundancy for web information extraction. Proc VLDB Endow **3**(1), 578–587 (2010)
12. Hu, S., Zou, L., Yu, J.X., Wang, H., Zhao, D.: (2018) Answering natural language questions by subgraph matching over knowledge graphs. In: 34th IEEE International Conference on Data Engineering, ICDE 2018. Paris, France, April 16-19, 2018, pp. 1815–1816
13. Kuang, J., Cao, Y., Zheng, J., He, X., Gao, M., Zhou, A.: Improving neural relation extraction with implicit mutual relations. In: 36th IEEE International Conference on Data Engineering, ICDE 2020. Dallas, TX, USA, April 20-24, 2020, pp. 1021–1032 (2020)
14. Li, Z., Sharaf, M.A., Sitbon, L., Du, X., Zhou, X.: Core: A context-aware relation extraction method for relation completion. IEEE Trans Knowl Data Eng **26**(4), 836–849 (2014)
15. Lin, Y., Shen, S., Liu, Z., Luan, H., Sun, M.: Neural relation extraction with selective attention over instances. In: Proceedings of ACL, pp. 2124–2133 (2016)
16. Liu, H., Li, Z., Sheng, D., Zheng, H., Shen, Y.: Multi-entity collabora-tive relation extraction. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021. Toronto, ON, Canada, June 6-11, 2021, IEEE, pp. 7678–7682 (2021)
17. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of ACL, pp. 1003–1011 (2009)
18. Miwa, M., Bansal, M.: End-to-end relation extraction using lstms on sequences and tree structures. In: Proceedings of ACL, pp. 1105–1116 (2016)
19. Nadgeri, A., Bastos, A., Singh, K., Mulang, I.O., Hoffart, J., Shekarpour, S., Saraswat, V.: Kgpool: Dynamic knowledge graph context selection for relation extraction. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021, Association for Computational Linguistics, Findings of ACL, vol ACL/IJCNLP 2021, pp. 535–548 (2021)
20. Qu, J., Ouyang, D., Hua, W., Ye, Y., Zhou, X.: Discovering correlations between sparse features in distant supervision for relation extraction. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019, pp. 726–734 (2019)
21. Reichartz, F., Korte, H., Paass, G.: Semantic relation extraction with kernels over typed dependency trees. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010, pp. 773–782 (2010)
22. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proceedings of EMNLP, pp. 148–163 (2010)
23. Sorokin, D., Gurevych, I.: Context-aware representations for knowledge base relation extraction. In: Proceedings of EMNLP, pp. 1784–1789 (2017)

24. Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C.D.: Multi-instance multi-label learning for relation extraction. In: Proceedings of EMNLP, pp. 455–465 (2012)
25. Vashishth, S., Joshi, R., Prayaga, S.S., Bhattacharyya, C., Talukdar, P.P.: RESIDE: improving distantly-supervised neural relation extraction using side information. In: Proceedings of EMNLP, pp 1257–1266 (2018)
26. Wang, H., Tan, M., Yu, M., Chang, S., Wang, D., Xu, K., Guo, X., Potdar, S.: Extracting multiple-relations in one-pass with pre-trained transformers. In: Korhonen, A., Traum, D.R., Màrquez L (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, Association for Computational Linguistics, pp. 1371–1377 (2019)
27. Wong, T., Lam, W.: Learning to adapt web information extraction knowledge and discovering new attributes via a bayesian approach. IEEE Trans Knowl Data Eng **22**(4), 523–536 (2010)
28. Wu, M., Pan, S., Zhu, X., Zhou, C., Pan, L.: Domain-adversarial graph neural networks for text classification. In: 2019 IEEE International Conference on Data Mining, ICDM 2019. Beijing, China, November 8-11, 2019, pp. 648–657 (2019)
29. Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., Ré C.: Fonduer: Knowledge base construction from richly formatted data. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018. Houston, TX, USA, June 10-15, 2018, pp. 1301–1316 (2018)
30. 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 EMNLP, pp. 1785–1794 (2015)
31. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Proceedings of COLING, pp. 2335–2344 (2014)
32. Zhang, Y., Qi, P., Manning, C.D.: Graph convolution over pruned dependency trees improves relation extraction. In: Proceedings of EMNLP, pp. 2205–2215 (2018)
33. Zhang, Y., Yao, Q., Shao, Y., Chen, L.: Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In: 35th IEEE International Conference on Data Engineering, ICDE 2019. Macao, China, April 8-11, 2019, pp. 614–625 (2019)
34. Zhu, H., Lin, Y., Liu, Z., Fu, J., Chua, T., Sun, M.: Graph neural networks with generated parameters for relation extraction. In: Proceedings of ACL, pp 1331–1339 (2019)