GraphIE: A Graph-Based Framework for Information Extraction

Anonymous authors

Abstract

Most modern Information Extraction (IE) systems are implemented as sequential taggers and only model local dependencies. Non-local and non-sequential context is, however, a valuable source of information to improve predictions. In this paper, we introduce GraphIE, a framework that operates over a graph representing a broad set of dependencies between textual units (i.e. words or sentences). The algorithm propagates information between connected nodes through graph convolutions, generating a richer representation that can be exploited to improve word-level predictions. Evaluation on three different tasks—namely social media, textual and visual information extraction—shows that GraphIE consistently outperforms the state-of-the-art sequence tagging model by a significant margin.1

1. Introduction

Most modern Information Extraction (IE) systems are implemented as sequential taggers. While such models effectively capture relations in the local context, they have limited capability of capturing non-local and non-sequential dependencies. In many applications, however, such dependencies can greatly reduce tagging ambiguity, thereby improving overall extraction performance. For instance, when extracting entities from a document, various types of non-local contextual information such as co-references and identical mentions may provide valuable cues. In Figure 1, we can see how such non-local relations are crucial to discriminate the entity type of the second mention of Washington (i.e. PERSON or LOCATION), which would instead be ambiguous if considered in isolation.

Most of the prior work looking at the non-local dependencies incorporates them by constraining the output space in a structured prediction framework [Finkel et al., 2005, Reichart and Barzilay, 2012, Hu et al., 2016]. Such approaches, however, have limited capacity to capture the richer set of structural relations in the input space. For instance, the co-referent dependencies shown in Figure 1 cannot be readily exploited for entity extraction.

1. Our code and data will be released at http://xxx.
He was involved in a wide range of duties for Washington’s request.

Figure 1: Example of the entity extraction task with an ambiguous entity mention (i.e. “...for Washington’s request...”). Aside from the sentential forward and backward edges (green) which aggregate local contextual information, non-local relations — such as the co-referent edges (red), the identical-mention edges (orange) and the subject connections across sentences (blue) — provide additional valuable information to reduce tagging ambiguity (i.e. Person or Location).

by constraining the output space, since the co-referent mentions are not necessarily labeled as entities. In the attempt to capture non-local dependencies in the input space, alternative approaches define a graph that outlines the input structure and engineer features to describe it [Quirk and Poon, 2017]. Designing effective features is however challenging, arbitrary and time consuming, especially when the underlying structure is complex. Moreover, these approaches have limited ability to capture node interactions informed by the graph structure.

In this paper, we propose GraphIE, a framework that improves predictions by automatically learning the interactions between local and non-local dependencies in the input space. Our approach integrates a graph module with the encoder-decoder architecture for sequence tagging. The algorithm operates over a graph, where nodes correspond to textual units (i.e. words or sentences) and edges describe their relations. At the core of our model, a recurrent neural network sequentially encodes local contextual representations and then the graph module iteratively propagates information between neighboring nodes using graph convolutions [Kipf and Welling, 2016]. The learned representations are finally projected back to a recurrent decoder to support tagging at the word level.

We evaluate GraphIE on three IE tasks, namely social media, textual, and visual [Aumann et al., 2006] information extraction. For each task, we provide in input a simple task-specific graph, which defines the data structure without accessing to any major processing or external resources. Our model is expected to learn from the relevant dependencies to identify and extract the appropriate information. Experimental results on multiple benchmark datasets show that GraphIE not only consistently outperforms a strong and commonly
adopted sequential model (SeqIE, i.e. a bi-directional long-short term memory (BiLSTM) followed by a conditional random fields (CRF) module), but it also obtains state-of-the-art performance among models not using external resources on the CoNLL03 dataset. More in details, in the social media IE task, GraphIE improves over SeqIE by 3.7% in extracting the Education attribute from twitter users. In the textual IE task, we obtain an improvement of 0.5% over SeqIE on the CoNLL03 dataset, and an improvement of 1.4% on the chemical entity extraction [Krallinger et al., 2015]. In visual IE, finally, we outperform the baseline by 1.2%.

2. Related Work

The problem of incorporating non-local and non-sequential context to improve information extraction has been extensively studied in the literature. The majority of methods have focused on enforcing constraints in the output space during inference, through various mechanisms such as posterior regularization or generalized expectations [Finkel et al., 2005, Mann and McCallum, 2010, Reichart and Barzilay, 2012, Li et al., 2013, Hu et al., 2016].

Research capturing non-local dependencies in the input space have mostly relied on feature-based approaches. [Roberts et al., 2008] and [Swampillai and Stevenson, 2011] have designed intra- and inter-sentential features based on discourse and syntactic dependencies (e.g., shortest paths) to improve relation extraction. [Quirk and Poon, 2017] used document graphs to flexibly represent multiple types of relations between words (e.g., syntactic, adjacency and discourse relations).

Graph-based representations can be also learned with neural networks. In this respect, the most related work to ours is the graph convolutional network by [Kipf and Welling, 2016], which was developed to encode graph structures and perform node classification. In our framework, we adapt GCN as an intermediate module that learns non-local context, which — instead of being used directly for classification — is projected to the decoder to enrich local information and perform sequence tagging.

A handful of other information extraction approaches have used graph-based neural networks. [Miwa and Bansal, 2016] applied Tree LSTM [Tai et al., 2015] to jointly represent sequences and dependency trees for entity and relation extraction. On the same line of work, [Peng et al., 2017, Song et al., 2018] introduced Graph LSTM, which extended the traditional LSTM to graphs by enabling a varied number of incoming edges at each memory cell. [Zhang et al., 2018] exploited graph convolutions to pool information over pruned dependency trees, outperforming existing sequence and dependency-based neural models in a relation extraction task. These studies differ from ours in several respects. First, they can only model word-level graphs, whereas our framework can learn non-local context either from word- or sentence-level graphs, using it to reduce ambiguity during tagging at the word level. Second, all these studies achieved improvements only when using dependency trees. We extend the graph-based approach to validate the benefits of using other types of relations in a broader range of tasks, such as co-reference in named entity recognition, followed-by link in social media, and layout structure in visual information extraction.
3. Problem Definition

We formalize information extraction as a sequence tagging problem. Rather than simply modeling inputs as sequences, we assume there exists a graph structure in the data that can be exploited to capture non-local and non-sequential dependencies between textual units, namely words or sentences.

We consider the input to be a set of sentences \( S = \{s_1, \ldots, s_N\} \) and an auxiliary graph \( G = (V, E) \), where \( V = \{v_1, \ldots, v_M\} \) is the node set and \( E \subset V \times V \) is the edge set. Each sentence is a sequence of words. We consider two different designs of the graph:

1. **sentence-level graph**, where each node is a sentence (i.e. \( M = N \)), and the edges encode sentence dependencies;

2. **word-level graph**, where each node is a word (i.e. \( M \) is the number of words in the input), and the edges connect pairs of words, such as co-referent tokens.

The edges \( e_{i,j} = (v_i, v_j) \) in the graph can be either directed or undirected. Multiple edge types can also be defined to capture different structural factors underlying the task-specific input data.

We use the BIO (Begin, Inside, Outside) tagging scheme in this paper. For each sentence \( s_i = (w_{1}^{(i)}, w_{2}^{(i)}, \ldots, w_{k}^{(i)}) \), we sequentially tag each word as \( y_i = (y_{1}^{(i)}, y_{2}^{(i)}, \ldots, y_{k}^{(i)}) \).

4. Method

GraphIE jointly learns local and non-local dependencies by iteratively propagating information between node representations. Our model has three components:

- **an encoder**, which generates local context-aware hidden representations for the textual unit (i.e. word or sentence, depending on the task) with a recurrent neural network;

- **a graph module**, which captures the graph structure and learns non-local and non-sequential dependencies between textual units;

- **a decoder**, which exploits the contextual information generated by the graph module to perform labelling at the word level.

Figure 2 illustrates the overview of GraphIE and the model architectures for both sentence- and word-level graphs. In the following sections, we first introduce the case of the sentence-level graph, and then we explain how to adapt the model for the word-level graph.

4.1 Encoder

In GraphIE, we first use an encoder to generate text representations. Given a sentence \( s_i = (w_{1}^{(i)}, w_{2}^{(i)}, \ldots, w_{k}^{(i)}) \) of length \( k \), we encode it with a recurrent neural network (RNN). Each word \( w_{t}^{(i)} \) is represented by a vector \( x_{t}^{(i)} \), which is the concatenation of its word embedding.

---

2. While sentences may have different lengths, for notation simplicity we use a single variable \( k \).
and a feature vector learned with a character-level convolutional neural network (CharCNN; [Kim et al., 2016]). We choose a bi-directional LSTM [Hochreiter and Schmidhuber, 1997] to process the sentence, defining it as

$$[h_1^{(i)}, \ldots, h_k^{(i)}] = \text{RNN} \left( [x_1^{(i)}, \ldots, x_k^{(i)}]; 0, \Theta_{\text{enc}} \right),$$

where $\Theta_{\text{enc}}$ represents the encoder parameters, and 0 indicates the initial hidden states of the BiLSTM encoder.

We obtain the sentence representation for $s_i$ by averaging the hidden states of its words, i.e. $\text{Enc}(s_i) = \frac{1}{k} \left( \sum_{t=1}^{k} h_t^{(i)} \right)$.

### 4.2 Graph Module

The graph module is designed to learn the non-local and non-sequential information from the graph. We adapt the graph convolutional network (GCN) to model the graph context for information extraction.
Given the sentence-level graph $G = (V, E)$, where each node $v_i$ (i.e. sentence $s_i$) has the encoding $\text{Enc}(s_i)$ capturing its local information, the graph module enriches such representation with neighbor information derived from the graph structure.

Our graph module is a GCN which takes as input the sentence representation, i.e. $g^{(0)}_i = \text{Enc}(s_i)$, and conducts graph convolution on every node, propagating information between its neighbors, and integrating such information into a new hidden representation. Specifically, each layer of GCN has two parts. The first gets the information of each node from the previous layer, i.e.

$$\alpha^{(l)}_i = W^{(l)}_v g^{(l-1)}_i,$$

where $W^{(l)}_v$ is the weight to be learned. The second aggregates information from the neighbors of each node, i.e. for node $v_i$, we have

$$\beta^{(l)}_i = \frac{1}{d(v_i)} \cdot W^{(l)}_e \left( \sum_{e_{i,j} \in E} g^{(l-1)}_j \right),$$

where $d(v_i)$ is the degree of node $v_i$ (i.e. the number of edges connected to $v_i$) and is used to normalize $\beta^{(l)}_i$, ensuring that nodes with different degrees have representations of the same scale. In the simplest case, where the edges in the graph are undirected and have the same type, we use the same weight $W^{(l)}_e$ for all of them. In a more general case, where multiple edge types exist, we expect them to have different impacts on the aggregation. Thus, we model these edge types with different weights in Eq. 3, similar to the relational GCN proposed by [Schlichtkrull et al., 2018]. When edges are directed, i.e. edge $e_{i,j}$ is different from $e_{j,i}$, the propagation mechanism should mirror such difference. In this case, we consider directed edges as two types of edges (forward and backward), and use different weights for them.

Finally, $\alpha^{(l)}_i$ and $\beta^{(l)}_i$ are combined to obtain the representation at the $l$-th layer,

$$g^{(l)}_i = \sigma \left( \alpha^{(l)}_i + \beta^{(l)}_i + b^{(l)} \right),$$

where $\sigma(\cdot)$ is the non-linear activation function, and $b^{(l)}$ is a bias parameter.

Because each layer only propagates information between directly connected nodes, we can stack multiple graph convolutional layers to get a larger receptive field, i.e. each node can be aware of more distant neighbors. After $L$ layers, for each node $v_i$ we obtain a contextual representation,

$$\text{GCN}(s_i) = g^{(L)}_i,$$

that captures both local and non-local information.

### 4.3 Decoder

To support tagging, the learned representation is propagated to the decoder.

In our work, the decoder is instantiated as a BiLSTM+CRF tagger [Lample et al., 2016]. The output representation of the graph module, $\text{GCN}(s_i)$, is split into two vectors of
the same length, which are used as the initial hidden states for the forward and backward
LSTMs, respectively. In this way, the graph contextual information is propagated to each
word through the LSTM. Specifically, we have

$$
[\mathbf{z}^{(i)}_1, \cdots, \mathbf{z}^{(i)}_k] = \text{RNN} \left( \left[ \mathbf{h}^{(i)}_1, \cdots, \mathbf{h}^{(i)}_k \right]; \text{GCN}(s_i), \Theta_{\text{dec}} \right),
$$

where $\mathbf{h}^{(i)}_t$ are the output hidden states of the encoder, $\text{GCN}(s_i)$ represents the initial state,
and $\Theta_{\text{dec}}$ is the decoder parameters. Finally, we use a CRF layer [Lafferty et al., 2001] on
top of the BiLSTM to perform tagging,

$$
y^*_i = \arg \max_{y \in \mathbf{Y}_k} p \left( y \left| [\mathbf{z}^{(i)}_1, \cdots, \mathbf{z}^{(i)}_k]; \Theta_{\text{crf}} \right. \right),
$$

where $\mathbf{Y}_k$ is the set of all possible tag sequences of length $k$, and $\Theta_{\text{crf}}$ represents the CRF
parameters, i.e. transition scores of tags. CRF combines the local predictions of BiLSTM
and the transition scores to model the joint probability of the tag sequence.\(^4\)

### 4.4 Adaptation to Word-level Graphs

GraphIE can be easily adapted to model word-level graphs. In such case, the nodes represent
words in the input, i.e. the number of nodes $M$ equals the total number of words in the $N$
sentences. At this point, each word’s hidden state in the encoder can be used as the input
node vector $\mathbf{g}^{(0)}_i$ of the graph module. GCN can then conduct graph convolution on the
word-level graph and generate graph-contextualized representations for the words. Finally,
the decoder directly operates on the GCN’s outputs, i.e. we change the BiLSTM decoder to

$$
[\mathbf{z}^{(i)}_1, \cdots, \mathbf{z}^{(i)}_k] = \text{RNN} \left( \left[ \text{GCN}(\mathbf{w}^{(i)}_1), \cdots, \text{GCN}(\mathbf{w}^{(i)}_k) \right]; 0, \Theta_{\text{dec}} \right),
$$

where $\text{GCN}(\mathbf{w}^{(i)}_t)$ is the GCN output for word $\mathbf{w}^{(i)}_t$. In this case, the BiLSTM initial states
are set to the default zero vectors. The CRF layer remains unchanged.

As it can be seen in Figure 2(c), the word-level graph module differs from the sentence-
level one because it directly takes the word representations from the encoder and feeds its
output to the decoder. In sentence-level graph, the GCN operates on sentence representa-
tions, which are then used as the initial states of the decoder BiLSTM.

### 5. Experimental Setup

We evaluate the model on three tasks, including two traditional IE tasks, namely social
media information extraction and textual information extraction, and an under-explored
task — visual information extraction. For each of these tasks, we created a simple task-
specific graph topology, designed to easily capture the underlying structure of the input
data without any major processing. Table 1 summarizes the characteristics of the three
tasks.

\(^4\) In GraphIE, the graph module models the input space structure, i.e. the dependencies between textual
units (i.e. sentences or words), and the final CRF layer models the sequential connections of the output
tags. Even though loops may exist in the input graph, CRF will operate sequentially, thus the inference
is tractable.
### Table 1: Comparisons of graph structure in the three IE tasks used for evaluation.

<table>
<thead>
<tr>
<th>Task</th>
<th>Graph Type</th>
<th>Node</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social Media IE</td>
<td>sentence-level</td>
<td>user’s tweets</td>
<td>followed-by</td>
</tr>
<tr>
<td>Textual IE</td>
<td>word-level</td>
<td>word</td>
<td>1. non-local consistency (identical mentions)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. local sentential forward and backward</td>
</tr>
<tr>
<td>Visual IE</td>
<td>sentence-level</td>
<td>text box</td>
<td>spatial layout (horizontal and vertical)</td>
</tr>
</tbody>
</table>

### Table 2: Statistics of the Education and Job datasets (Task 1).

<table>
<thead>
<tr>
<th></th>
<th>Education</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>7,208</td>
<td>1,772</td>
</tr>
<tr>
<td>Edges</td>
<td>11,167</td>
<td>3,498</td>
</tr>
<tr>
<td>Positive Tweets</td>
<td>49,793</td>
<td>3,694</td>
</tr>
<tr>
<td>Negative Tweets</td>
<td>393,683</td>
<td>172,349</td>
</tr>
</tbody>
</table>

### 5.1 Task 1: Social Media Information Extraction

Social media information extraction refers to the task of extracting information from users’ posts in online social networks [Benson et al., 2011, Li et al., 2014]. In this paper, we aim at extracting education and job information from users’ tweets. Given a set of tweets posted by a user, the goal is to extract mentions of the organizations to which they belong. The fact that the tweets are short, highly contextualized and show special linguistic features makes this task particularly challenging.

**Dataset**  We construct two datasets, Education and Job, from the Twitter corpus released by [Li et al., 2014]. The original corpus contains millions of tweets generated by ≈ 10 thousand users, where the education and job mentions are annotated using distant supervision [Mintz et al., 2009]. We sample the tweets from each user, maintaining the ratio between positive and negative posts. The obtained Education dataset consists of 443,476 tweets generated by 7,208 users, and the Job dataset contains 176,043 tweets generated by 1,772 users. Dataset statistics are reported in Table 2.

The datasets are both split in 60% for training, 20% for development, and 20% for testing. We perform 5 different random splits and report the average results.

**Graph Construction**  We construct the graph as ego-networks [Leskovec and Mcauley, 2012], i.e. when we extract information about one user, we consider the subgraph formed by the user and his/her direct neighbors. Each node corresponds to a Twitter user, who is represented by the set of posted tweets. Edges are defined by the followed-by link, under

---

5. Positive and negative refer here to whether or not the education or job mention is present in the tweet. In [Li et al., 2014], sampling was not necessary because they processed the tweets and extracted engineered features beforehand, and then learned the model.

6. As each node is a set of tweets posted by the user, we encode every tweet with the encoder, and then average them to obtain the node representation. In the decoding phase, the graph module’s output is fed to the decoder for each tweet.
the assumption that connected users are more likely to come from the same university or company. An example of Education social media graph is reported in the Supplemental Material.

5.2 Task 2: Textual Information Extraction

In this task, we focus on named entity recognition at discourse level (DiscNER). In contrast to traditional sentence-level NER (SentNER), where sentences are processed independently, in DiscNER, long-range dependencies and constraints across sentences have a crucial role in the tagging process. For instance, multiple mentions of the same entity are expected to be tagged consistently in the same discourse. Here we propose to utilize this (soft) consistency constraint to improve entity extraction.

Dataset

We conduct experiments on two NER datasets: the CoNLL-2003 shared task dataset (CoNLL03) [Tjong et al., 2003] and the Chemdner dataset for chemical entity extraction [Krallinger et al., 2015]. We follow the standard split of each corpora. Statistics are shown in Table 3.

Graph Construction

In this task, nodes represent words. We create two types of edges for each given document:

- **Local edges**: forward and backward edges are created between neighboring words in each sentence, allowing local contextual information to be utilized.

- **Non-local edges**: re-occurrences of the same token other than stop words are connected, so that information can be propagated through, encouraging global consistency of tagging.\(^7\)

5.3 Task 3: Visual Information Extraction

Visual information extraction refers to the extraction of attribute values from documents formatted in various layouts. Examples include invoices and forms, whose format can be exploited to infer valuable information to support extraction.

---

7. Note that other non-local relations such as co-references and subject connections (cf. the example in Figure 1) may be used for further improvement. However, these relations require additional resources or modules to obtain, and we leave them to future work.
Dataset  The corpus consists of 25,200 Adverse Event Case Reports (AECR) recording drug-related side effects. Each case contains an average of 9 pages. Since these documents are produced by multiple organizations, they exhibit large variability in the layout and presentation styles (e.g. text, table, etc.). The collection is provided with a separate human-extracted ground truth database that is used as a source of distant supervision.

Our goal is to extract eight attributes related to the patient, the event, the drug and the reporter (cf. Table 6 for the full list). Attribute types include dates, words and phrases — which can be directly extracted from the document.

The dataset is split in 50% cases for training, 10% for development, and 40% for testing.

Graph Construction  We first turn the PDFs to text using PDFMiner, which provides words along with their positions in the page (i.e. bounding-box coordinates). Consecutive words are then geometrically joined into text boxes. Each text box is considered as a “sentence” in this task, and corresponds to a node in the graph.

Because the page layout is the major structural factor in these documents, we work on page-by-page basis, i.e. each page corresponds to a graph. The edges are defined to horizontally or vertically connect nodes (text boxes) that are close to each other (i.e. when the overlap of their bounding boxes, in either the vertical or horizontal direction, is over 50%). Four types of edge are considered: left-to-right, right-to-left, up-to-down, and down-to-up. When multiple nodes are aligned, only the closest ones are connected. An example of visual document graph is reported in the Supplemental Material.

5.4 Baseline and Our Method

We implement a two-layer BiLSTM with a conditional random fields (CRF) tagger as the sequential baseline (SeqIE). This architecture and its variants have been extensively studied and demonstrated to be successful in previous work on information extraction [Lample et al., 2016, Ma and Hovy, 2016]. In the textual IE task (Task 2), our baseline is shown to obtain competitive results with the state-of-the-art method in the CONLL03 dataset. In the visual IE task (Task 3), in order to further increase the competitiveness of the baseline, we sequentially concatenate the horizontally aligned text boxes, therefore fully modeling the horizontal edges of the graph.

Our baseline shares the same encoder and decoder architecture with GraphIE, but without the graph module. In Task 1 and Task 3, we apply GraphIE with sentence-level graph module (cf. Figure 2(b)), and in Task 2, we apply GraphIE with word-level graph module (cf. Figure 2(c)).

5.5 Implementation Details

The models are trained with Adam [Kingma and Ba, 2014] to minimize the CRF objective. For regularization, we choose dropout with a ratio of 0.1 on both the input word representation and the hidden layer of the decoder. The learning rate is set to 0.001. We use the development set for early-stopping and the selection of the best performing hyperparameters. For CharCNN, we use 64-dimensional character embeddings and 64 filters of

---

8. This dataset cannot be shared for patient privacy and proprietary issues.
Table 4: Extraction accuracy on the Education and Job datasets (Task 1). Scores are the average of 5 runs. * indicates the improvement over SeqIE is statistically significant (Welch's t-test, $p < 0.01$).

6. Results

6.1 Task 1: Social Media Information Extraction

Table 4 shows the results for the social media information extraction task. GraphIE outperforms SeqIE in both the Education and Job datasets, and the improvements are more significant for the Education dataset (3.7% versus 0.3%). The reason for such difference is the variance in the affinity scores [Mislove et al., 2010] between the two datasets. [Li et al., 2014] underline that affinity value for Education is 74.3 while for Job it is only 14.5, which means that in the datasets neighbors are 5 times more likely to have studied in the same university than worked in the same company. We can therefore expect that a model like GraphIE, which exploits neighbors’ information, obtains larger advantages in a dataset characterized by higher affinity.

6.2 Task 2: Textual Information Extraction

Table 5 describes the NER accuracy on the CoNLL03 [Tjong et al., 2003] and the ChEMD-NER [Krallinger et al., 2015] datasets.

For CoNLL03, we list the performance of existing approaches. As it can be seen, our baseline SeqIE already obtains competitive scores compared to the best methods. The fact that GraphIE significantly outperforms it, establishing a new state-of-the-art performance on the CoNLL03 dataset, highlights once more the importance of modeling non-local and non-sequential dependencies and confirms that our approach is an appropriate method to achieve this goal.

10. Here we refer to the best performance of methods not using external resources. Further improvement can be achieved by exploiting contextualized embeddings pretrained on large corpora [Peters et al., 2018, Devlin et al., 2018].
## 6.3 Task 3: Visual Information Extraction

Table 6 shows the results in the visual information extraction task. GraphIE outperforms the SeqIE baseline in most attributes, and achieves 1.2% improvement in the micro average F1 score. It confirms that the benefits of using layout graph structure in visual information extraction.

The extraction performance varies across the attributes, ranging from 61.4% for Drug Name to 95.8% for Patient Date of Birth (similar variations are visible in the baseline).
GraphIE: A Graph-Based Framework for Information Extraction

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphIE</td>
<td>77.8</td>
</tr>
<tr>
<td>– Edge types</td>
<td>77.0 (↓ 0.8)</td>
</tr>
<tr>
<td>– Horizontal edges</td>
<td>74.7 (↓ 3.1)</td>
</tr>
<tr>
<td>– Vertical edges</td>
<td>72.4 (↓ 5.4)</td>
</tr>
<tr>
<td>– CRF</td>
<td>72.1 (↓ 5.7)</td>
</tr>
</tbody>
</table>

Table 7: Ablation study (Task 3). Scores are micro average F1 on the development set. “↓” means removing the element from GraphIE.

Similarly, the gap between GraphIE and SeqIE varies in relation to the attributes, ranging between −0.5% in Patient Date of Birth and 2.4% in Event.

In the ablation test described in Table 7, we can see the contribution of: using separate weights for different edge types (+0.8%), horizontal edges (+3.1%), vertical edges (+5.4%), and CRF (+5.7%).

**Generalization** We also assess GraphIE’s capacity of dealing with unseen layouts through an extra analysis. From our dataset, we sample 2,000 reports containing the three most frequent templates, and train the models on this subset. Then we test all models in two settings: 1) *seen templates*, consisting of 1,000 additional reports in the same templates used for training; and 2) *unseen templates*, consisting of 1,000 reports in two new template types.

The performance of GraphIE and SeqIE is reported in Figure 3. Both models achieve good results on *seen templates*, with GraphIE still scoring 2.8% higher than SeqIE. The gap becomes even larger when our model and the sequential one are tested on *unseen templates* (i.e. 20.3%), demonstrating that by explicitly modeling the richer structural relations, GraphIE achieves better generalizability.

### 7. Conclusions

We introduced GraphIE, a framework for learning local and non-local contextual representations from graph structures. The system operates over a task-specific graph topology, designed to easily capture the underlying structure of the input data without any major processing. GraphIE jointly models the node (i.e. textual units, namely words or sentences) representations and their dependencies. Graph convolutions project information through neighboring nodes to finally support the decoder during tagging at the word level.

We evaluated our framework on three IE tasks, namely social media, textual and visual information extraction. Our results show that supporting predictions with non-local and non-sequential context consistently enhances accuracy and outperforms the competitive SeqIE baseline (based on a two-layer BiLSTM followed by a CRF).

Future work includes the exploration of computationally sustainable ways to automatically learn the underlying graphical structure of the input data.
References


Figure 1: Mock-up example of Education Social Media Information Extraction (Task 1). Nodes are represented as sequences of tweets and edges are follow-by relations.

Figure 2: Mock-up example of two forms with different layouts (Task 3). Graphical dependencies are shown as green lines connecting text in blue bounding-boxes.