PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks

*Wenwen Yu†, *Ning Lu‡, Xianbiao Qi‡, Ping Gong† and Rong Xiao‡
†School of Medical Imaging, Xuzhou Medical University, Xuzhou, China
‡Visual Computing Group, Ping An Property & Casualty Insurance Company, Shenzhen, China
Email: yuwenwen62@gmail.com, gongping@xzhmu.edu.cn
{jiangxiluning, qixianbiao, rongxiao}@gmail.com
* *denotes equal contribution*

***Abstract*—Computer vision with state-of-the-art deep learning models has achieved huge success in the field of Optical Character Recognition (OCR) including text detection and recognition tasks recently. However, Key Information Extraction (KIE) from documents as the downstream task of OCR, having a large number of use scenarios in real-world, remains a challenge because documents not only have textual features extracting from OCR systems but also have semantic visual features that are not fully exploited and play a critical role in KIE. Too little work has been devoted to efficiently make full use of both textual and visual features of the documents. In this paper, we introduce PICK, a framework that is effective and robust in handling complex documents layout for KIE by combining graph learning with graph convolution operation, yielding a richer semantic representation containing the textual and visual features and global layout without ambiguity. Extensive experiments on real-world datasets have been conducted to show that our method outperforms baselines methods by significant margins. Our code is available at https://github.com/wenwenyu/PICK-pytorch.**

I. Introduction

Computer vision technologies with state-of-the-art deep learning models have achieved huge success in the field of OCR including text detection and text recognition tasks recently. Nevertheless, KIE from documents as the downstream task of OCR, compared to typical OCR tasks, had been a largely under explored domain and is also a challenging task [1]. The aim of KIE is to extract texts of a number of key fields from given documents, and save the texts to structured documents. KIE is essential for a wide range of technologies such as efficient archiving, fast indexing, document analytics and so on, which has a pivotal role in many services and applications.

Most KIE systems simply regard extraction tasks as a sequence tagging problems and implemented by Named Entity Recognition (NER) [2] framework, processing the plain text as a linear sequence result in ignoring most of valuable visual and non-sequential information (e.g., text, position, layout, and image) of documents for KIE. The main challenge faced by many researchers is how to fully and efficiently exploit both textual and visual features of documents to get a richer semantic representation that is crucial for extracting key information without ambiguity in many cases and the expansibility of the method [3]. See for example Figure 1(c), in which the layout and visual features are crucial to discriminate the entity type of TOTAL. Figure I shows different layouts and types of documents.

The traditional approaches use hand-craft features (e.g., regex and template matching) to extract key information as shown in Figure 2(a). However, this solution [4], [5] only uses text and position information to extract entity and need a large amount of task-specific knowledge and human-designed rules, which does not extend to other types of documents. Most modern methods considered KIE as a sequence taggers problem and solved by NER as shown in Figure 2(b). In comparison to the typical NER task, it is much more challenging to distinguish entity without ambiguity from complicated documents for a machine. One of the main reasons is that such a framework only operates on plain texts and not corporates visual information and global layout of documents to get a richer representation. Recently, a few studies in the task of KIE have attempted to make full use of untapped features in complex documents. [6] proposed LayoutLM method, inspired by BERT [7], for document image understanding using pre-training of text and layout. Although this method uses image features and position to pre-train model and performs well on downstream tasks for document image understanding such as KIE, it doesn't consider the latent relationship between two text segments. Besides, this model needs adequate data and time consuming to pre-train model inefficiently.

Alternative approaches [8], [9] predefine a graph to combine textual and visual information by using graph convolutions operation [10] as illustrated in Figure 2(c). In the literature of [8], [9], the relative importance of visual features and non-sequential information is debated and graph neural networks modeling on document brings well performance on extraction entity tasks. But [8] needs prior knowledge and extensive human efforts to predefine task-specific *edge* type and *adjacent matrix* of the graph. Designing effective edge type and adjacent matrix of the graph, however, is challenging, subjective, and

(a) Medical invoice (b) Train ticket (c) Tax receipt

Figure 1. Examples of documents with different layouts and types.

time-consuming, especially when the structure of documents is sophisticated. [9] directly define a fully connected graph then uses a self-attention mechanism to define convolution on fully connected nodes. This method probably ignores the noise of the node and leads to aggregate useless and redundancy node information.

In this paper, we propose **PICK**, a robust and effective method shown in Figure 2(d), **P**rocessing Key **I**nformation Extraction from Documents using improved Graph Learning-**C**onvolutional Networ**K**s, to improve extraction ability by automatically making full use of the textual and visual features within documents. PICK incorporates *graph learning* module inspired by [11] into existing graph architecture to learn a *soft adjacent matrix* to effectively and efficiently refine the graph context structure indicating the relationship between nodes for downstream tasks instead of predefining edge type of the graph artificially. Besides, PICK make full use of features of the documents including text, image, and position features by using *graph convolution* to get richer representation for KIE. The graph convolution operation has the powerful capacity of exploiting the relationship generated by the graph learning module and propagates information between nodes within a document. The learned richer representations are finally used to a decoder to assist sequence tagging at the character level. PICK combines a graph module with the encoder-decoder framework for KIE tasks as illustrated in Figure 2(d).

The main contributions of this paper can be summarized as follows:

Brief Summary

- In this paper, we present a novel method for KIE, which is more effective and robust in handling complex documents layout. It fully and efficiently uses features of documents (including text, position, layout, and image) to get a richer semantic representation that is crucial for extracting key information without ambiguity.
- We introduce improved graph learning module into the model, which can refine the graph structure on the complex documents instead of predefining struct of the graph.
- Extensive experiments on real-world datasets have been conducted to show that our method outperforms baselines methods by significant margins.

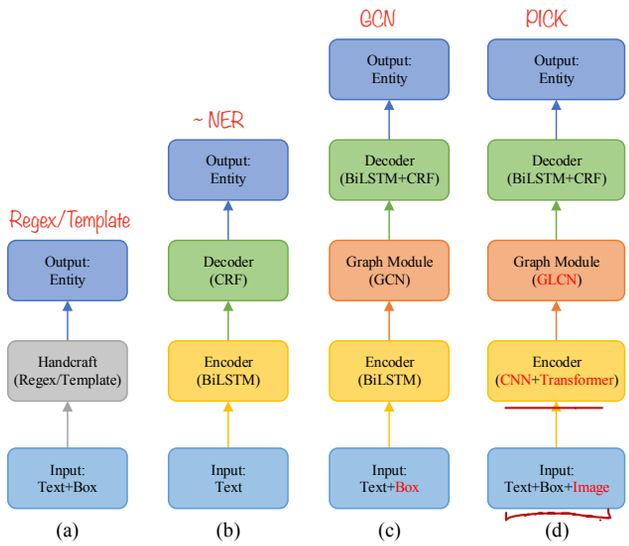

Figure 2. Typical architectures and our method for key information extraction. (a) hand-craft features based method. (b) automatic extraction features based method. (c) using more richer features based method. (d) our proposed models.

II. Related Work

Existing research recognizes the critical role played by making full use of both textual and visual features of the documents to improve the performance of KIE. The majority of methods, however, pay attention to the textual features, through various features extractors such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs) on word- and character- level [2], [12], [13]. Although [14], [15] uses visual image features to process extraction, it only focuses on image features and does not take textual features into account. [6] attempts to use both textual and visual features for document understanding and gets good performance on some documents, through pre-training of text and layout, but it doesn't consider the relationship between text within documents. Besides, a handful of other methods [16], [17], [18] make full use of features to support extraction tasks based on human-designed features or task-specific knowledge, which are not extensible on other documents.

Besides, recent research using both textual and visual features to aid the extraction mainly depends on graph-based representations due to graph convolutional networks (GCN) [10]

Figure 2
a. Regex, Template Matching : Non intelligent way.
b. NER : Does not account into visual features a lot, only relies on text and relative positions of word tokens.
c. Text + Graph based visual features : GCN training hard, tuning and designing of adjacent matrix is complicated.
d. Text + new graph learning module based visual features : soft adjacent matrix to make GCN training easier (Proposed).

GCN Types (Need more Information)

a. Spatial : defines graph Conv operations as operations directly on node group of neighbours. Aggregating features from neighbours.

b. Spectral : Introduces filters for graph processing according to graph spectral theory. Drawback: entire graph to be processed simultaneously.

demonstrated huge success in unstructured data tasks. Overall, GCN methods can be split into spatial convolution and spectral convolution methods[19]. The graph convolution our framework used to get a richer representation belongs to the spatial convolution category which generally defines graph convolution operation directly through defining an operation on node groups of neighbors [20], [21]. Spectral methods that generally define graph convolution operation based on the spectral representation of graphs[10], however, are not propitious to dynamic graph structures. [22], [23] proposed Graph LSTM, which enables a varied number of incoming dependencies at each memory cell. [24] jointly extract entities and relations through designing a directed graph schema. [25] proposes a version of GCNs suited to model syntactic dependency graphs to encode sentences for semantic role labeling. [26] proposed a lexicon-based GCN with global semantics to avoid word ambiguities. Nevertheless, their methods don't take visual features into the model.

A few Methods Without Visual Features

The most related works to our method are [8], [9], using graph module to capture non-local and multimodal features for extraction but still differ from ours in several aspects. First, [8] only use textual and position features where images are not used and need to predefine task-specific edge type and connectivity between nodes of the graph. Nevertheless, our method can automatically learn the relationship between nodes by graph learning module, using it to efficiently refine the structure of the graph without any prior knowledge in order to aggregate more useful information by graph convolution. Second, [9] also does not use images features to improve the performance of extraction tasks without ambiguity. Meanwhile, due to [9] simply and roughly regards graph as fully connectivity no matter how complicated the documents are, graph convolution aggregates useless and redundancy information between nodes. Our method, however, incorporating graph learning into framework, can filter useless nodes and be robust to document complex layout structure.

III. Method

In this section, we provide a detailed description of our proposed method, PICK. The overall architecture is shown in Figure 3, which contains 3 modules:

- *Encoder*: This module encodes text segments using Transformer to get text embeddings and image segments using CNN to get image embeddings. The text segments and image segments stand for textual and morphology information individually. Then these two types of embeddings are combined into a new local representation $\mathbf{X}$, which will be used as node input to the Graph Module.
- *Graph Module*: This module can catch the latent relation between nodes and get richer graph embeddings representation of nodes through improved graph learning-convolutional operation. Meanwhile, bounding boxes containing layout context of the document are also modeled into the graph embeddings so that graph module can get non-local and non-sequential features.
- *Decoder*: After obtaining the graph embeddings of the document, this module performs sequence tagging on the union non-local sentence at character-level using BiLSTM and CRF, respectively. In this way, our model transforms key information extraction tasks into a sequence tagging problem by considering the layout information and the global information of the document.

To ease understanding, our full model is described in parts. First, we begin by introducing the notation used in this paper in Section III-A. Our encoder representation is described in Section III-B and the proposed graph module mechanism is then described in Section III-C. Finally, Section III-D shows how to combine the graph embedding and text embedding to output results.

A. Notation

Given a document $\mathcal{D}$ with N sentences/text segments, its representation is denoted by $S = \{s_1, \ldots, s_N\}$, where s_i is a set of characters for the i-th sentence/text segments. We denote s_i^{is} and s_i^{bb} as image segments and bounding box at position i, respectively. For each sentence $s_i = (c_1^{(i)}, \ldots, c_T^{(i)})$, we label each character as $\mathbf{y}_i = (y_1^{(i)}, \ldots, y_T^{(i)})$ sequentially using the IOB (Inside, Outside, Begin) tagging scheme [27], where T is the length of sentence s_i.

An accessory graph of a document $\mathcal{D}$ is denoted with $G = (V, R, E)$, where $V = \{v_1, \ldots, v_N\}$ is a set of N nodes, $R = \{\alpha_{i1}, \ldots, \alpha_{ij}\}$, α_{ij} is the set of relations between two nodes, and $E \subset V \times R \times V$ is the edge set and each edge $e_{ij} = (v_i, \alpha_{ij}, v_j) \in E$ represents that the relation $\alpha_{ij} \in R$ exist from node v_i to v_j.

B. Encoder

As shown in Figure 3, the top-left position in the diagram is the encoder module, which contains two branches. Different from the existing key information works [8], [9] that only use text segments or bounding boxes, one of our key contribution in this paper is that we also use image segments simultaneously containing morphology information to improve document representations performance which can be exploited to help key information extraction tasks.

One branch of Encoder generates text embeddings using encoder of Transformer [28] for capturing local textual context. Given a sentence $s_i = (c_1^{(i)}, \ldots, c_T^{(i)})$, text embeddings of sentence s_i is defined as follows

$$\mathbf{te}_{1:T}^{(i)} = \texttt{TransformerEncoder}\left(\mathbf{c}_{1:T}^{(i)}; \Theta_{\text{tenc}}\right), \quad (1)$$

where $\mathbf{c}_{1:T}^{(i)} = [\mathbf{c}_1^{(i)}, \ldots, \mathbf{c}_T^{(i)}]^T \in \mathbb{R}^{T \times d_{model}}$ denotes the input sequence, $\mathbf{c}_t^{(i)} \in \mathbb{R}^{d_{model}}$ represents a token embedding (e.g., Word2Vec) of each character $c_t^{(i)}$, d_{model} is the dimension of the model, $\mathbf{te}_{1:T}^{(i)} = [\mathbf{te}_1^{(i)}, \ldots, \mathbf{te}_T^{(i)}]^T \in \mathbb{R}^{T \times d_{model}}$ denotes the output sequence, $\mathbf{te}_t^{(i)} \in \mathbb{R}^{d_{model}}$ represents the encoder output of Transformer for the i-th character $c_t^{(i)}$, and Θ_{tenc} represents the encoder parameters of Transformer. Each sentence is encoded independently and we can get a document $\mathcal{D}$ text embeddings, defining it as

$$\mathbf{TE} = [\mathbf{te}_{1:T}^{(1)}; \ldots; \mathbf{te}_{1:T}^{(N)}] \in \mathbb{R}^{N \times T \times d_{model}}. \quad (2)$$

Modules

- Encoder : Image segments Embeddings (CNN) and Text segments Embeddings (Transformer) are combined to form a node.
- Graph Module : Captures relation b/w nodes by graph Conv operations. Bbox of Layout content are also modelled into graph embeddings (How is this done?)
- Decoder : performs sequence tagging on union, non local sentences at character level using BiLSTM and CRF. (Sequence tagging task but on graph embeddings)

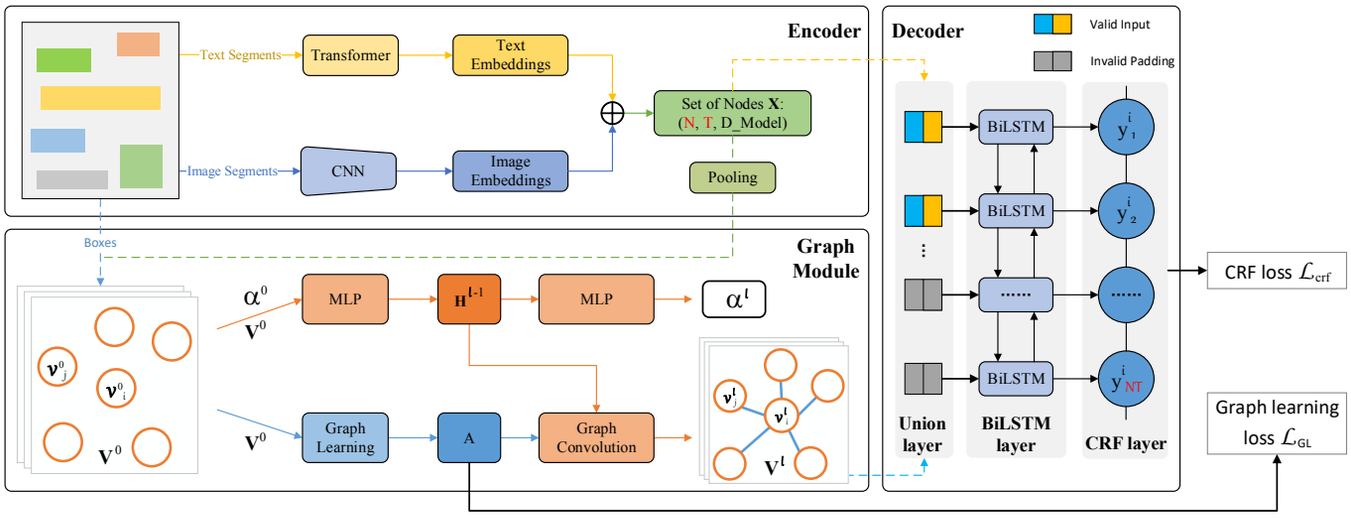

Figure 3. Overview of PICK. $\mathbf{V}^l$ donates *node embedding* in the l-th graph convolution layer. $\boldsymbol{\alpha}^l$ and $\mathbf{H}^l$ represents *relation embedding* and *hidden features* between the node v_i and v_j in the l-th graph convolution layer, respectively. $\mathbf{A}$ is *soft adjacent matrix*. N, T, and D_Model denotes the number of sentences segments, the max-length of sentence and the dimension of the model respectively. $\oplus$ denotes element-wise addition.

Another branch of Encoder generate image embedding using CNN for catching morphology information. Given a image segment s_i^{is}, image embeddings is defined as follows

$$\mathbf{ie}^{(i)} = \texttt{CNN}\left(\mathbf{s_i}^{is}; \Theta_{\text{cnn}}\right), \tag{3}$$

where $\mathbf{s_i}^{is} \in \mathbb{R}^{H' \times W' \times 3}$ denotes the vector of input image segment, H' and W' represent the height and width of image segment s_i^{is} respectively, $\mathbf{ie}^{(i)} \in \mathbb{R}^{H \times W \times d_{model}}$ represents the output of CNN for the i-th image segment s_i^{is}, and Θ_{cnn} represents the CNN parameters. We implement the CNN using ResNet [29] and resize image under condition $H \times W = T$ then encode each image segments individually and we can get a document $\mathcal{D}$ image embeddings, defining it as

$$\mathbf{IE} = [\mathbf{ie}^{(1)}; \ldots; \mathbf{ie}^{(N)}] \in \mathbb{R}^{N \times T \times d_{model}}. \tag{4}$$

Finally, we combine text embeddings $\mathbf{TE}$ and image embeddings $\mathbf{IE}$ through element-wise addition operation for feature fusion and then generate the fusion embeddings $\mathbf{X}$ of the document $\mathcal{D}$, which can be expressed as

Element wise addition

$$\mathbf{X} = \mathbf{TE} + \mathbf{IE}, \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{N \times T \times d_{model}}$ represent a set of nodes of graph ant $\mathbf{X}$ will be used as input $\mathbf{X}_0$ of Graph Module followed by pooling operation and $\mathbf{X}_0 \in \mathbb{R}^{N \times d_{model}}$.

C. Graph Module

Existing key information works [8], [9] using graph neural networks modelling global layout context and non-sequential information need prior knowledge to pre-define task-specific *edge* type and *adjacent matrix* of the graph. [8] define *edge* to horizontally or vertically connected nodes/text segments that are close to each other and specify four types of *adjacent matrix* (left-to-right, right-to-left, up-to-down, and down-to-up). But this method cannot make full use of all graph nodes and excavate latent connected nodes that are far apart in the document. Although [9] use a fully connected graph that every node/text segments is connected, this operation leads to graph aggregate useless and redundancy node information.

In this way, we incorporate improved graph *learning-convolutional network* inspired by [11] into existing graph architecture to learn a *soft adjacent matrix* $\mathbf{A}$ to model the graph context for downstream tasks illustrated in the lower left corner of Figure 3.

1) Graph Learning: Given an input $\mathbf{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]^T \in \mathbb{R}^{N \times d_{model}}$ of graph nodes, where $\boldsymbol{v}_i \in \mathbb{R}^{d_{model}}$ is the i-th node of the graph and the initial value of $\mathbf{V}$ is equal to $\mathbf{X}_0$, Graph Module generate a *soft adjacent matrix* $\mathbf{A}$ that represents the pairwise relationship weight between two nodes firstly through graph learning operation, and extract features $\mathbf{H}$ for each node v_i using a multi-layer perception (MLP) networks just like [9] on input $\mathbf{V}$ and corresponding relation embedding $\boldsymbol{\alpha}$. Then we perform graph convolution operation on features $\mathbf{H}$, propagating information between nodes and aggregate such information into a new feature representation $\mathbf{V}'$. Mathematically, we learn a *soft adjacent matrix* $\mathbf{A}$ using a single-layer neural work as

$$\begin{cases} \mathbf{A}_i = \text{softmax}(\mathbf{e}_i), \quad i = 1, \ldots, N, \quad j = 1, \ldots, N, \\ \mathrm{e}_{ij} = \text{LeakRelu}(\mathbf{w}_i^T |\boldsymbol{v}_i - \boldsymbol{v}_j|), \end{cases} \tag{6}$$

where $\mathbf{w}_i \in \mathbb{R}^{d_{model}}$ is learnable weight vector. To solve the problem of gradients vanishing at training phase, we use LeakRelu instead of Relu activation function. The function $\text{softmax}(\cdot)$ is conducted on each row of $\mathbf{A}$, which can guarantee that the learned *soft adjacent matrix* $\mathbf{A}$ can satisfy the following property

$$\sum\nolimits_{j=1}^{N} A_{ij} = 1, A_{ij} \geq 0. \tag{7}$$

We use the modified loss function based on [11] to optimize the learnable weight vector $\mathbf{w}_i$ as follows

$$\mathcal{L}_{\text{GL}} = \frac{1}{N^2}\sum_{i,j=1}^{N} \exp(A_{ij} + \eta\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2) + \gamma\|\mathbf{A}\|_F^2\,, \quad (8)$$

where $\|\cdot\|_F$ represents Frobenius-Norm. Intuitively, the first item means that nodes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are far apart in higher dimensions encouraging a smaller weight value A_{ij}, and the exponential operation can enlarge this effect. Similarly, nodes that are close to each other in higher dimensional space can have a stronger connection weight. This process can prevent graph convolution aggregating information of noise node. η is a tradeoff parameter controlling the importance of nodes of the graph. We also average the loss due to the fact that the number of nodes is dynamic on the different documents. The second item is used to control the sparsity of *soft adjacent matrix* $\mathbf{A}$. γ is a tradeoff parameter and larger γ brings about more sparsity *soft adjacent matrix* $\mathbf{A}$ of graph. We use $\mathcal{L}_{\text{GL}}$ as a regularized term in our final loss function as shown in Eq.(17) to prevent trivial solution, i.e., $\mathbf{w}_i = \mathbf{0}$ as discussed in [11].

2) Graph Convolution: Graph convolutional network (GCN) is applied to capture global visual information and layout of nodes from the graph. We perform graph convolution on the *node-edge-node* triplets (v_i, α_{ij}, v_j) as used in [9] rather than on the node v_i alone.

Firstly, given an input $\mathbf{V}^0 = \mathbf{X}_0 \in \mathbb{R}^{N\times d_{model}}$ as the initial layer input of the graph, initial *relation embedding* $\boldsymbol{\alpha}_{ij}^0$ between the node v_i and v_j is formulated as follows

$$\boldsymbol{\alpha}_{ij}^0 = \mathbf{W}_\alpha^0[x_{ij}, y_{ij}, \frac{w_i}{h_i}, \frac{h_j}{h_i}, \frac{w_j}{h_i}, \frac{T_j}{T_i}]^T\,, \quad (9)$$

where $\mathbf{W}_\alpha^0 \in \mathbb{R}^{dmodel\times 6}$ is learnable weight matrix. x_{ij} and y_{ij} are horizontal and vertical distance between the node v_i and v_j respectively. w_i, h_i, w_j, h_j are the width and height between the node v_i and v_j individually. $\frac{w_i}{h_i}$ are the aspect ratio of node v_i, and $\frac{h_j}{h_i}, \frac{w_j}{h_i}$ uses the height of the node v_i for normalization and has affine invariance. Different from [9], we also use the sentences length ratio $\frac{T_j}{T_i}$ between the node v_i and v_j. Intuitively, the length of sentence contains latent importance information. For instance, in medical invoice, the age value entity is no more than three digits usually, which plays a critical role in improving key information extraction performance. Moreover, given the length of sentence and image, model can infer rough font size of text segments, which makes *relation embedding* get more richer representation.

Then we extract *hidden features* $\mathbf{h}_{ij}^l$ between the node v_i and v_j from the graph using the *node-edge-node* triplets (v_i, α_{ij}, v_j) data in the l-th convolution layer, which is computed by

$$\boldsymbol{h}_{ij}^l = \sigma(\mathbf{W}_{v_ih}^l\mathbf{v}_i^l + \mathbf{W}_{v_jh}^l\mathbf{v}_j^l + \boldsymbol{\alpha}_{ij}^l + \mathbf{b}^l)\,, \quad (10)$$

where $\mathbf{W}_{v_ih}^l, \mathbf{W}_{v_jh}^l \in \mathbb{R}^{d_{model}\times d_{model}}$ are the learnable weight matrices in the l-th convolution layer, and $\mathbf{b}^l \in \mathbb{R}^{dmodel}$ is a bias parameter. $\sigma(\cdot) = \max(0,\cdot)$ is an non-linear activation function. *Hidden features* $\boldsymbol{h}_{ij}^l \in \mathbb{R}^{d_{model}}$ represent the sum of visual features and the relation embedding between the node v_i and v_j which is critical to aggregate more richer representation for downstream task.

Finally, *node embedding* $\mathbf{v}_i^{l+1}$ aggregate information from *hidden features* $\boldsymbol{h}_{ij}^l$ using graph convolution to update node representation. As graph learning layer can get an optimal adaptive graph *soft adjacent matrix* $\mathbf{A}$, graph convolution layers can obtain task-specific *node embedding* by conducting the layer-wise propagation rule. For node v_i, we have

$$\mathbf{v}_i^{(l+1)} = \sigma(\mathbf{A}_i\mathbf{h}_i^l\mathbf{W}^l)\,, \quad (11)$$

where $\mathbf{W}^l \in \mathbb{R}^{d_{model}\times d_{model}}$ is layer-specific learnable weight matrix in the l-th convolution layer, and $\mathbf{v}_i^{(l+1)} \in \mathbb{R}^{d_{model}}$ donates the node embedding for node v_i in the $l+1$-th convolution layer. After L layers, we can get a contextual information $\mathbf{v}_i^L$ containing global layout information and visual information for every node v_i. Then $\mathbf{v}_i^L$ is propagated to the decoder for tagging task.

The *relation embedding* $\boldsymbol{\alpha}_{ij}^{l+1}$ in the $l+1$-th convolution layer for node v_i is formulated as

$$\boldsymbol{\alpha}_{ij}^{l+1} = \sigma(\mathbf{W}_\alpha^l\mathbf{h}_{ij}^l)\,, \quad (12)$$

where $\mathbf{W}_\alpha^l \in \mathbb{R}^{d_{model}\times d_{model}}$ is layer-specific trainable weight matrix in the l-th convolution layer.

D. Decoder

The decoder shown in Figure 3 consists of Union layer, BiLSTM [30] layer and CRF [31] layer for key information extraction. *Union layer* receives the input $\mathbf{X} \in \mathbb{R}^{N\times T\times d_{model}}$ having variable length T generated from Encoder, then packs padded input sequences and fill padding value at the end of sequence yielding packed sequence $\hat{\mathbf{X}} \in \mathbb{R}^{(N\cdot T)\times d_{model}}$. Packed sequence $\hat{\mathbf{X}}$ can be regarded as the union non-local document representation instead of local text segments representation when performs sequence tagging using CRF. Besides, We concatenated the *node embedding* of the output of Graph Module to packed sequence $\hat{\mathbf{X}}$ at each timestamps. Intuitively, node embedding containing the layout of documents and contextual features as auxiliary information can improve the performance of extraction without ambiguity. BiLSTM can use both past/left and future/right context information to form the final output. The output of BiLSTM is given by

$$\mathbf{Z} = \texttt{BiLSTM}\left(\hat{\mathbf{X}}\,; \mathbf{0}, \Theta_{\text{lstm}}\right)\mathbf{W}_z\,, \quad (13)$$

where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N\cdot T}]^{N\cdot T} \in \mathbb{R}^{(N\cdot T)\times d_{output}}$ is the output of BiLSTM and denotes the scores of emissions matrix, d_{output} is the number of different entity, $\mathbf{Z}_{t,j}$ represents the score of the j-th entity of the t-th character c_t in packed sequence $\hat{\mathbf{X}}$, $\mathbf{0}$ means the initial hidden state and is zero, and Θ_{lstm} represents the BiLSTM parameters. $\mathbf{W}_z \in \mathbb{R}^{d_{model}\times d_{output}}$ is the trainable weight matrix.

Given a packed sequence $\hat{\mathbf{X}}$ of predictions $\mathbf{y}$, its scores can be defined as follows

$$s(\hat{\mathbf{X}}, \mathbf{y}) = \sum_{i=0}^{N\cdot T} T_{y_i, y_{i+1}} + \sum_{i=1}^{N\cdot T} \mathbf{Z}_{i,y_i}\,, \quad (14)$$

where $\mathbf{T} \in \mathbb{R}^{(N\cdot T+2)\times(N\cdot T+2)}$ is the scores of transition matrix and $\mathbf{y} = (y_1, \ldots, y_{N\cdot T})$. y_0 and $y_{N\cdot T+1}$ represent the 'SOS' and 'EOS' entity of a sentence, which means start of sequence and end of sequence respectively. $T_{i,j}$ represents the score of a transition from the entity i to entity j.

Then the sequence CRF layer generates a family of conditional probability via a softmax for the sequence $\mathbf{y}$ given $\hat{\mathbf{X}}$ as follows

$$p(\mathbf{y}|\hat{\mathbf{X}}) = \frac{e^{s(\hat{\mathbf{X}},\mathbf{y})}}{\sum_{\widetilde{\mathbf{y}}\in\mathcal{Y}(\hat{\mathbf{X}})} e^{s(\hat{\mathbf{X}},\widetilde{\mathbf{y}})}}, \tag{15}$$

where $\mathcal{Y}(\hat{\mathbf{X}})$ is all possible entity sequences for $\hat{\mathbf{X}}$.

For CRF training, we minimize the negative log-likelihood estimation of the correct entity sequence and is given by

$$\begin{cases} \mathcal{L}_{\text{crf}} = -\ \log(p(\mathbf{y}|\hat{\mathbf{X}})) = -s(\hat{\mathbf{X}},\mathbf{y}) + Z\,, \\ Z = \log\left(\sum_{\widetilde{\mathbf{y}}\in\mathcal{Y}(\hat{\mathbf{X}})} e^{s(\hat{\mathbf{X}},\widetilde{\mathbf{y}})}\right) = \underset{\widetilde{\mathbf{y}}\in\mathcal{Y}(\hat{\mathbf{X}})}{\text{logadd}}\ s(\hat{\mathbf{X}},\widetilde{\mathbf{y}})\,. \end{cases} \tag{16}$$

Our model parameters of whole networks are jointly trained by minimizing the following loss function as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{crf}} + \lambda\mathcal{L}_{\text{GL}}\,, \tag{17}$$

where $\mathcal{L}_{\text{GL}}$ and $\mathcal{L}_{\text{crf}}$ are defined in Eq. 8 and Eq. 16 individually, and λ is a tradeoff parameter.

Decoding of CRF layer is to search the output sequence $\mathbf{y}^*$ having the highest conditional probability

$$\mathbf{y}^* = \underset{\widetilde{\mathbf{y}}\in\mathcal{Y}(\hat{\mathbf{X}})}{\text{argmax}}\, p(\widetilde{\mathbf{y}}|\hat{\mathbf{X}})\,. \tag{18}$$

Training (Eq. 16) and decoding (Eq. 18) phase are time-consuming procedure but we can use the dynamic programming algorithm to improve speed.

IV. Experiments

A. Datasets

Medical Invoice is our collected dataset containing 2,630 images. It has six key text fields including medical insurance type, Chinese capital total amount, invoice number, social security number, name, and hospital name. This dataset mainly consists of digits, English characters, and Chinese characters. An example of an anonymized medical invoice is shown in Fig. 1(a). The medical invoice is variable layout datasets with the illegible text and out of position print font. For this dataset, 2,104 and 526 images are used for training and testing individually.

Train Ticket contains 2k real images and 300k synthetic images proposed in [14]. Every train ticket has eight key text fields including ticket number, starting station, train number, destination station, date, ticket rates, seat category, and name. This dataset mainly consists of digits, English characters, and Chinese characters. An example of an anonymized train ticket image is shown Fig. 1(b). The train ticket is fixed layout dataset, however, it contains background noise and imaging distortions. The datasets do not provide text bounding boxes (bbox) and the transcript of each text bbox. So we randomly selected 400 real images and 1,530 synthetic images then human annotate bbox and use the OCR system to get the transcript of each text bbox. For the annotated dataset, all our selected synthetic images and 320 real images are used for training and the rest of real images for testing.

SROIE [1] contains 626 receipts for training and 347 receipts for testing. Every receipt has four key text fields consisting of company, address, date, and total. This dataset mainly contains digits and English characters. An example receipt is shown in Fig. 1(c), and this dataset have a variable layout with a complex structure. The SROIE dataset provides text bbox and the transcript of each text bbox.

B. Implementation Details

Networks Setting In the encoder part, the text segments feature extractor is implemented by the encoder module of Transformer [28] yielding text embeddings and the image segments feature extractor is implemented by ResNet50 [29] generating image embeddings. The hyper-parameter of Transformer used in our paper is same as [28] produced outputs of dimension $d_{model} = 512$. Then the text embeddings and image embeddings are combined by element-wise addition operation for feature fusion and then as the input of the graph module and decoder. The graph module of the model consists of graph learning and graph convolution. In our experiments, the default value of η, γ in graph learning loss is $1, 0.4$ individually and the number of the layer of graph convolution $L = 1$. The decoder is composed of BiLSTM [30] and CRF [31] layers. In the BiLSTM layer, the hidden size is set to 512, and the number of recurrent layers is 2. The tradeoff parameter of training loss λ is 0.01 in the decoder.

Evaluation Metrics In the medical invoice, train ticket, and SROIE scenario, in the condition of a variable number of appeared entity, mean entity recall (mER), mean entity precision (mEP), and mean entity F-1 (mEF) defined in [14] are used to benchmark performance of PICK.

Label Generation For train ticket datasets, we annotated the bbox and label the pre-defined entity type for each bbox then use the OCR system to generate the transcripts corresponding to bbox. When we get bbox and the corresponding entity type and transcripts of bbox, we enumerate all transcripts of the bbox and convert it to IOB format [27] used to minimize CRF loss. For SROIE datasets, due to the fact that it only provides bbox and transcripts, so we annotated the entity type for each bbox, then the rest of the operation is the same as the train ticket. Different from train ticket and SROIE datasets that directly use human-annotated entity type to generate IOB format label, we adopted a heuristic approach provided in [9] to get the label of Medical Invoice because human-annotated bbox probably cannot precisely match OCR system detected box in the process of practical application.

Implementation The proposed model is implemented in PyTorch and trained on 8 NVIDIA Tesla V100 GPUs with 128 GB memory. Our model is trained from scratch using Adam [49] as the optimizer to minimize the CRF loss and graph learning loss jointly and the batch size is 16 at the training phase. The learning rate is set to 10^{-4} over the whole training

Datasets:
- Medical Invoices: 2630 (2104/526) documents, 6 fields.
- Train Ticket: 2k real 300k synthetic, 8 fields. 400 real and 1530 synthetic images are annotated manually(bbox) (1530+320/80). Fixed layout.
- SROIE: 626/347, 4 fields, bbox + text provided

Evaluation Metrics:
- meR: mean entity recall
- meP: mean Entity Precision
- meF: mean entity F1 score

Encoder: Transformer(dims=512)+Resnet50(dims=512)
Graph Module:: GCN Layers =1,
Decoder: recurrent layers=2, hidden layer size=512

Table I
PERFORMANCE COMPARISON BETWEEN PICK (OURS) AND BASELINE METHOD ON MEDICAL INVOICE DATASETS. PICK IS MORE ACCURATE THAN THE BASELINE METHOD. **BOLD** REPRESENT THE BEST PERFORMANCE.

Entities	Baseline			PICK (Our)		
	mEP	mER	mEF	mEP	mER	mEF
Medical Insurance Type	66.8	77.1	71.6	85.0	81.1	**83.0**
Chinese Capital Total Amount	85.7	88.9	87.3	93.1	98.4	**95.6**
Invoice Number	61.1	57.7	59.3	93.9	90.9	**92.4**
Social Security Number	53.4	64.6	58.5	71.3	64.6	**67.8**
Name	73.1	73.1	73.1	74.7	85.6	**79.8**
Hospital Name	69.3	74.4	71.8	78.1	89.9	**83.6**
Overall (micro)	71.1	73.4	72.3	85.0	89.2	**87.0**

Is baseline just BiLSTM+CRF (NER) setup or is it GCN set up in paper [8] [9]??

- Most benefits seen for fields in different colours or visibly different font type.
- Does well for fixed layouts as expected.
- Datasets tested on are very small and hard to say how well it will do on complex datasets.
- Only incremental gains from image segments and graph learning module

phase. We also use dropout with a ratio of 0.1 on both BiLSTM and the encoder of the Transformer. Our model is trained for 30 epochs, each epoch takes about 35 minutes. At the inference phase, the model directly predicts every text segment which belongs to the most possible entity type without any post-processed operation or constraint rules to correct the results except for SROIE. For the task of extraction of SROIE, we use a lexicon which is built from the train data to autocorrect results.

Baseline method To verify the performance of our proposed method, we apply a two-layer BiLSTM with a CRF tagger to the baseline method. This architecture has been extensively proved and demonstrated to be valid in previous work on KIE [8], [9]. All text segments of documents are concatenated from left to right and from top to bottom yielding a one-dimensional textual context as the input of the baseline to execute extraction tasks. The hyper-parameter of BiLSTM of the baseline is similar to the PICK method.

Table II
RESULTS COMPARISON ON SROIE AND TRAIN TICKET DATASETS. THE EVALUATION METRIC IS MEF.

Method	Train Ticket	SROIE
Baseline	85.4	-
LayoutLM [6]	-	95.2
PICK (Ours)	98.6	96.1

Table III
RESULTS OF EACH COMPONENT OF OUR MODEL. THE EVALUATION METRIC IS MEF.

Model	Medical Invoice	Train Ticket
PICK (Full model)	87.0	98.6
w/o image segments	↓0.9	↓0.4
w/o graph learning	↓1.6	↓0.7

C. Experimental Results

We report our experimental results in this section. In the medical invoice scenario, as can be seen from the Table I, the average mEF scores of baseline and PICK were compared to verify the performance of the PICK. What is striking about the figures in this table is that PICK outperforms the baseline in all entities, and achieves 14.7% improvement in the overall mEF score. Further analysis showed that the most striking aspect of the data is the biggest increase in Invoice Number mEF performance. Note that Invoice Number has distinguishing visual features with red color fonts than other text segments as shown in the top left corner of Fig. 1(a). In summary, these results show that the benefits of using both visual features and layout structure in KIE.

Bold Results

Furthermore, see from the second column of Table II, PICK shows significant improvement over the baseline method in the train ticket scenario. Surprisingly, the mEF of PICK almost get a full score on the train ticket. This result suggests that PICK can handle very well extraction tasks on fixed layout documents due to PICK having the ability to learn the graph structure of documents. We also use online evaluation tools[1] on SROIE datasets to verify our competitive performance. As we can see from the third column of Table II, our model achieves competitive results in mEF metrics in condition of only using the training data provided by official. Note that LayoutLM [6] also uses extra pre-training datasets and documents class supervised information to train the model. Data from this table can be compared with the data in Table I which shows the robustness of our model on both variable and fixed layout documents.

D. Ablation Studies

In order to evaluate the contributions of each component of our model, we perform ablation studies in this section. As described in Table III, when we remove image segments element from PICK, the most striking observation to emerge from the data comparison is the drop in performance of PICK on both medical invoice and train ticket datasets. This indicates that visual features can play an important role in addressing the issue of ambiguously extracting key information. This result is not counter-intuitive as image segments can provide richer appearance and semantic features such as font colors, background, and directions. Additionally, image segments can help the graph module capture a reasonable graph structure. Furthermore, the improved graph learning module also makes a difference in the performance of the PICK. More specifically, as shown in Table III, removing graph learning element from PICK leads to a large metrics score cut down on two datasets, especially on variable layout datasets. Thus, graph learning can deal with not only the fixed layout but also variable layout datasets. So graph learning element is good at dealing with the complex structures of documents and generalization.

[1]https://rrc.cvc.uab.es/?ch=13&com=evaluation&task=3

Table IV
PERFORMANCE COMPARISONS OF DIFFERENT GRAPH CONVOLUTION LAYERS FOR DIFFERENT DATASETS.THE EVALUATION METRIC IS MEF.

Configuration	Medical Invoice	Train Ticket
$L = 1$	87.0	**98.6**
$L = 2$	**87.1**	97.2
$L = 3$	85.9	96.5
$L = 4$	85.34	92.8

This is slightly peculiar, does it mean the dataset used was small and prone to overfitting due to low complexity

We perform another ablation studies to analyze the impact of the different number of layers L of graph convolution on the extraction performance. As shown in Table IV, all best results are obtained with a 1- or 2-layer model rather than a 3- or 4-layer model. This result is somewhat counter-intuitive but this phenomenon illustrates a characteristic of the GCN [10] that the deeper the model (number of layers) is, the more it probably will be overfitting. In practice, we should set a task-specific number of layers of the graph.

This doesn't add up, why should this be task specific and not data specific

V. CONCLUSIONS

In this paper, we study the problem of how to improve KIE ability by automatically making full use of the textual and visual features within documents. We introduce the improved graph learning module into the model to refine the graph structure on the complex documents given visually rich context. It shows superior performance in all the scenarios and shows the capacity of KIE from documents with variable or fixed layout. This study provides a new perspective on structural information extraction from documents.

REFERENCES

[1] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. Jawahar, "ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 1516–1520.

[2] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2016, p. 260270.

[3] Y. Aumann, R. Feldman, Y. Liberzon, B. Rosenfeld, and J. Schler, "Visual information extraction," *Knowledge and Information Systems*, vol. 10, no. 1, pp. 1–15, 2006.

[4] D. Schuster, K. Muthmann, D. Esser, A. Schill, M. Berger, C. Weidling, K. Aliyev, and A. Hofmeier, "Intellix–end-user trained information extraction for document archiving," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 101–105.

[5] A. Simon, J.-C. Pret, and A. P. Johnson, "A fast algorithm for bottom-up document layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 273–277, 1997.

[6] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of Text and Layout for Document Image Understanding," *arXiv preprint arXiv:1912.13318*, 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019.

[8] Y. Qian, E. Santus, Z. Jin, J. Guo, and R. Barzilay, "GraphIE: A graph-based framework for information extraction," in *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[9] X. Liu, F. Gao, Q. Zhang, and H. Zhao, "Graph convolution for multimodal information extraction from visually rich documents," in *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[10] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[11] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 313–11 320.

[12] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[13] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[14] H. P. Guo, X. Qin, J. Liu, J. Han, J. Liu, and E. Ding, "Eaten: Entity-aware attention for single shot visual text extraction," in *15th International Conference on Document Analysis and Recognition*, 2019.

[15] A. R. Katti, C. Reisswig, C. Guder, S. Brarda, S. Bickel, J. Höhne, and J. B. Faddoul, "Chargrid: Towards understanding 2d documents," *arXiv preprint arXiv:1809.08799*, 2018.

[16] K. Swampillai and M. Stevenson, "Extracting relations within and across sentences," in *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, 2011, pp. 25–32.

[17] M. Rusinol, T. Benkhelfallah, and V. Poulain dAndecy, "Field extraction from administrative documents by incremental structural templates," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1100–1104.

[18] F. Lebourgeois, Z. Bublinski, and H. Emptoz, "A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents," in *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. II. Conference B: Pattern Recognition Methodology and Systems*. IEEE, 1992, pp. 272–276.

[19] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: Algorithms, applications and open challenges," in *International Conference on Computational Social Networks*. Springer, 2018, pp. 79–91.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[21] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[22] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "N-ary relation extraction using graph state lstm," *arXiv preprint arXiv:1808.09101*, 2018.

[23] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[24] S. Wang, Y. Zhang, W. Che, and T. Liu, "Joint extraction of entities and relations based on a novel graph scheme." in *IJCAI*, 2018, pp. 4461–4467.

[25] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," *arXiv preprint arXiv:1703.04826*, 2017.

[26] T. Gui, Y. Zou, Q. Zhang, M. Peng, J. Fu, Z. Wei, and X.-J. Huang, "A lexicon-based graph neural network for chinese ner," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 1039–1049.

[27] E. F. Sang and J. Veenstra, "Representing text chunks," in *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 173–179.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 770–778.

[30] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2047–2052.

[31] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. ICML*, 2001.