Key Information Extraction from Vietnamese Invoices by Combining Layout and Context

Capstone Project Report

Ngo Tuan Anh
Tran Manh Cuong

A thesis submitted in part fulfilment of the degree of BSc. in Computer Science with the supervision of M.S. Le Dinh Huynh

Bachelor of Computer Science
Hoa Lac Campus – FPT University
April 27, 2021

Copyright © 2021

Ngo Tuan Anh & Tran Manh Cuong

All Rights Reserved

ACKNOWLEDGEMENT

Firstly, we would like to thank our instructor, Le Dinh Huynh, for his patience and time and instructing and advising us enthusiastically. Secondly, we would also like to thank our University, FPT, for giving us an excellent environment to study and grow over the years. Finally, we always remember our family's encouragement and support. We want to provide them with a special thanks because they motivate us to improve ourselves every day.

PROJECT SPECIFICATION

Tran Manh Cuong

- Research on methods of key information extraction (LayoutLM, PICK-pytorch).
- Write code to crawl data for VAT invoices (name, address, company name, tax, etc.).
- Write documentation (Introduction, Related work, Dataset).

Ngo Tuan Anh

- Research on methods of key information extraction (LayoutLM, PICK-pytorch).
- Merge the data with the image to generate a complete invoice.
- Implement the model in google colaboratory.
- Write documentation (Proposal methodology, Results, Conclusion, Reference).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis introduces a deep approach, an effective and robust framework in handling complex document layout, visual features, and textual semantics for Key Information Extraction (KIE). The algorithm combines graph learning with graph convolution, resulting in a richer semantic representation that includes both textual and visual features and a clear global layout. The model's input only with the coordinates of token bounding boxes, avoiding the use of raw images. It leads to a layout-aware language model, which can fine-tune downstream tasks. The model is evaluated on a key information extraction task using publicly available datasets SROIE. We show that it achieves superior performance on datasets consisting of visually rich documents while outperforming the baseline RoBERTa on documents.

Keywords: Key information extraction, Natural Language Processing, Layout analysis

CHAPTER 1. INTRODUCTION

Currently, the state-of-the-art Deep Learning models in Computer Vision have been very successful with Text Detection and Text Recognition. But there is a new problem that has not been fully explored and challenging: Key Information Extraction (KIE). Key information extraction(KIE) is extracting key information from textual sources to enable finding entities and classifying. There is also semantic advanced information extraction (also known as semantic annotation) that associates those entities with their descriptions and semantics from a knowledge graph.

Text data in general, invoices in particular, contain a lot of information, but not all information is important to you that you want to know. Think about when you have to look at all your bills to see how much to pay. Or go through the literature to find information about a problem. There are many other examples like collecting records, documents, analyzing reports, meeting minutes, billing, and more.

But imagine you have to view all that data manually and extract the necessary information. Sure, it's a very daunting, painful job, and you might even miss out on some critical information or make a mistake. It could have disastrous consequences for you or the business where you work.

One of the complex tasks when analyzing text is to find the correct information from the documents. Understand the primary content of the paper, or find hidden information. This is what anyone wants when they come across a piece of text. Therefore, finding out how to automatically extract data from textual data will reap many benefits and significantly reduce the time we spend skimming through text documents. This is what KIE wants to achieve. KIE's task extracts meaningful information from unstructured text data. It presents it in a structured format such as address, name, company, quant from various form types to formatted manner.

In business, an invoice is a document sent by the supplier to the buyer that lists information such as the seller, the seller's address, the tax code, the buyer's name, the address of the buyer or products purchased, when and how they were purchased. Large companies may receive tens to hundreds of thousands of invoices per year. They have used software and systems to manage such large volumes of bills. Many software and systems can only record the time of issuing and receiving invoices, control the number of invoices, detect duplicate invoices, flag or mark suspicious invoices, censor invoices with the order, etc. All to help the company manage their import and export. However, all information in the invoice cannot be extracted or imported into the software. Works like this are all done by humans. However, this is a very tedious, time-consuming, and also very costly job when hiring a lot of staff to extract invoices and import data into the system, so an exact solution. Automatic information will be of great benefit.

In this thesis, we used Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks, a powerful and effective method.

- With Text + Box + Image input, using CNN and Transformer to encode, using Graph Module as Graph Learning-Convolutional Network (GLCN), BiLSTM, and CRF to decode and obtain the output as entities. Processing Extracting Key Information from Documents Using Improved Graph Learning Convolutional NetworKs, to improve part extraction capabilities by automatically using full-text features and images in documents. Model incorporates graph-inspired learning module Semi-supervised learning with graph learning-convolutional networks into the existing graph architecture to learn a soft adjacent matrix to efficiently and effectively refine the graph context structure that shows the relationships between nodes for downstream tasks instead of defining artificially predetermined edge types.
- Besides, the model uses document features including text, image, and position features using graph convolution for richer representation for KIE. The graphical convolution operation can exploit the relationship created by the graph learning module and pass information between nodes in a document. Richer graphics learned are eventually used for the decoder to support character-level sequence tagging. The model combines the graph module with the encoder-decoder framework for KIE tasks.

The main contributions can be summarized as follows:

- We have contributed billing data.
- We have extracted KIE in Vietnamese invoice.

CHAPTER 2. RELATED WORK

Existing research recognizes the critical role that textual and visual features of the documents play in improving KIE performance.

2.1. Previous methods of improving KIE

However, on the word and character level, the majority of methods use different feature extractors such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to focus on textual features. An example is Neural Architectures for Named Entity Recognition (NER)**[1]**. In the above article, the authors have presented the neural architecture for the NER not to use language-specific resources. Models are designed to capture two intuitions. The first is because names usually consist of multiple tokens, so it is essential to inference together about tagging for each token. They compared two models, a two-dimensional LSTM model with a random conditional class from above (LSTM-CRF) and a new model that built and labeled chunks of input sentences equal to usage of the transition-based parsing algorithm with states represented by the LSTM (S-LSTM) stacks. The second, token-level evidence for “being a name,” including orthographic evidence (what does the word being tagged as a name look like?) and distributional evidence (where does the comment being tagged tend to occur in a corpus?). To capture orthographic sensitivity, used model character-based word representation to capture distributional sensitivity.

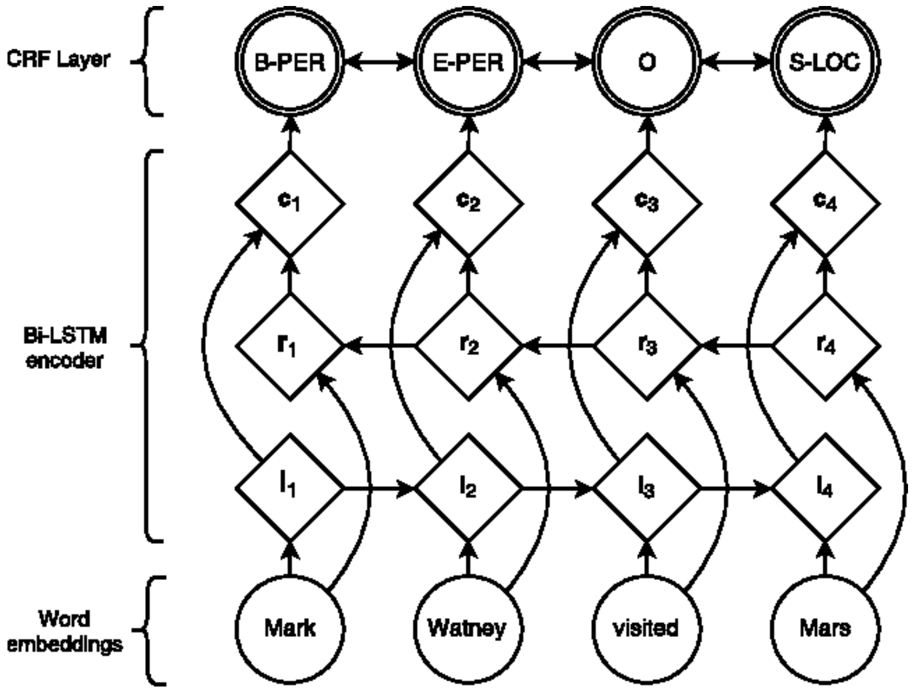


Figure 2.1. Main architecture of the network. Word embeddings are given to a bidirectional LSTM. li represents the word i and its left context, ri represents the word i and its right context. Concatenating these two vectors yields a representation of the word i in its context, ci **[1]**

Next is End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF **[2]**. In the paper, a neural network architecture is proposed for sequence labeling. This is an end-to-end model that requires no specific resources for the task, feature engineering, or data preprocessing, in addition to embedding the pre-trained word on the unlabeled corpus. Therefore, the model can be easily applied to a wide range of sequence labeling tasks across different languages and domains. The accumulation neural network (CNN) is first used to encode a word's character-level information into its character-level representation. Then combine letter and word-level representations, bringing them into a two-dimensional LSTM (BLSTM) to model the contextual information for each word. On top of BLSTM, use sequential CRF to co-decode labels for whole sentences.

Named Entity Recognition with Bidirectional LSTM-CNNs**[3]**, the above article presented a model combining two-dimensional LSTM and CNNs learning both character-level and word-level features, offering the first assessment of an ant. Such structure on the English language assessment dataset has been well established. Furthermore, since vocabulary is crucial to NER performance, they have proposed a new vocabulary encoding scheme and match algorithm that can use partial matches.

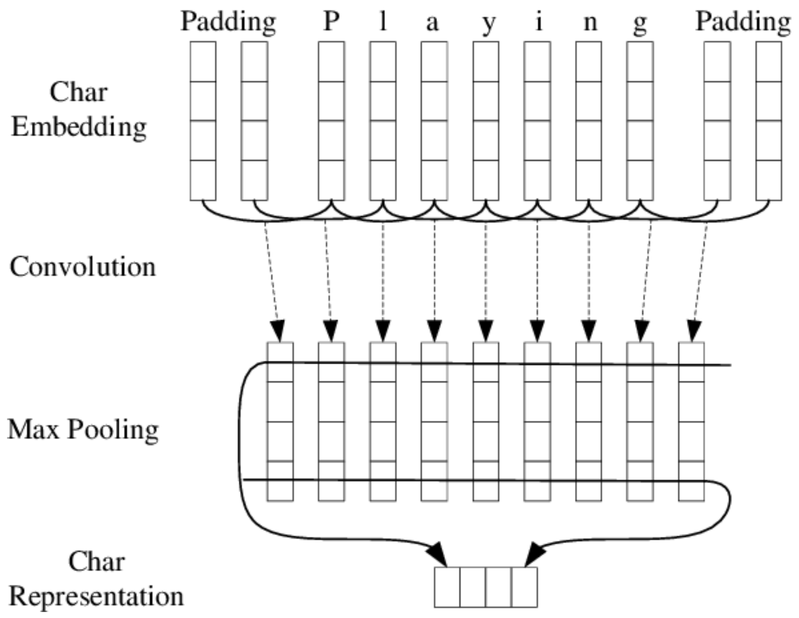


Figure 2.2. The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN **[2]**.

Although it uses visual image features to handle extraction, it only focuses on image features and does not consider text features such as the Eaten: Entity-aware attention for single-shot visual text extraction**[4]**. The above post uses a CNN-based feature extractor to extract feature maps from the original image, then design the entity-aware attention network, including multiple entity-aware decoders, initial state warm-up, and state transition between decoders to capture all entities in the image. EATEN can cover most of the corner cases with arbitrary shapes, projective/affine transformations, position drift without any correction due to introducing a spatial attention mechanism.

The next method is Chargrid: Towards understanding 2d documents **[5]**. In the above method a model for processing and understanding structured documents is introduced. Instead of serializing the document to 1D text, the proposed approach represents it as a sparse 2D grid of characters, which retains the document's spatial structure. The model predicts a segmentation mask with pixel level labels and object bound boxes to group multiple instances of the same layer. Try to use both text and images to understand the document and get good performance on some documents, through pre-training in text and layout, but it doesn't consider the relationship between text Documentation such as LayoutLM: Pre-training of Text and Layout for Document Image Understanding **[6]**. The above method is a pre-training but effective method for text and layout for document image comprehension tasks. Inspired by the BERT model, in which the input text information is mainly represented by embedding the text and embedding the position, LayoutLM adds two types of input embedding: embedding the 2-D position. display the relative position of the token within a document; embed the image for the scanned token image in the document. They added two ways of embedding this input because embedding a 2-D location can capture the relationship between tokens in a document, while embedding an image can capture some of the features that appear. like font orientation, type and color. Besides applying a multitasking learning target to LayoutLM, including Masked Visual-Language Model (MVLM) loss and Multi-label Document Classification (MDC) loss, which further enforces pre-training for text and layout. In this work, the focus is on pre-training documents based on the image of the scanned document, while digital documents are less challenging as they can be considered a special case without OCR.

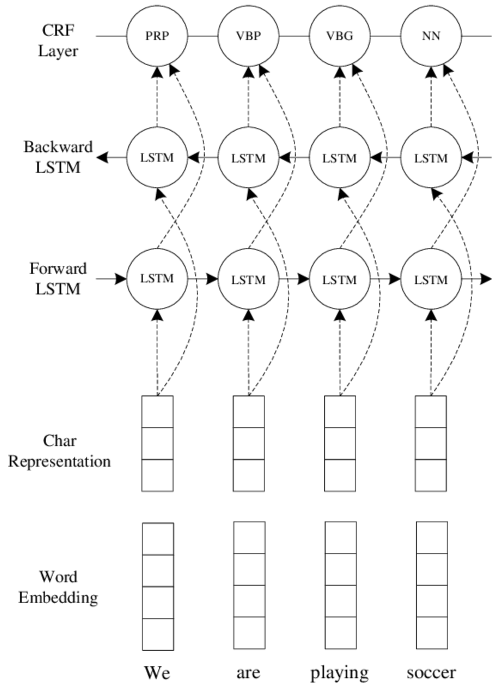


Figure 2.3. The main architecture of neural networks. The character representation for each word is computed by the CNN in Figure 2.1. Then the character representation vector is concatenated with the word embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on both the input and output vectors of BLSTM**[3]**.

Besides, some other methods make full use of features to support extraction tasks based on human-designed features or task-specific knowledge, which cannot be extended on other documents like Extracting relations within and across sentences **[7]**. Support Vector Machine extraction of both inter- and intra-sentential relations is used in this process. Some challenges to face when extracting inter-sentential relations. The structured features, which are based on the parser tree, have been used successfully to extract internal relations and do not naturally apply to many sentences. Solve this problem by introducing new structured features for the inter-sentential case. There is also a larger data sparsity problem when learning the exact models for inter-sentential relations partly because of the smaller number of denoted inter-sent connections. There is a learning approach called threshold adjustment (Shanahan and Roma, 2003 **[42]**) to counteract imbalances in the data.

The following method is Field extraction from administrative documents by incremental structural templates **[8]**. In the above method, the task is to extract relevant structured information from semi-structured document images. Suppose we focus on the specific case of invoice processing. In that case, the mission's purpose is to provide images coming from a known supplier so that all data can be extracted and stored in the database—related fields like date, total amount, invoice number, etc. There are several methods based on the definition of fixed spatial templates that map the locations where the OCR has to read the particular fields to extract. Such basic strategies can work perfectly in simple situations and cause problems right away when faced with large-scale situations and if document layout changes over time. Research papers often suggest structured patterns that encode the local context of information to be extracted. Such approaches learn the local layout structure from a sample document which is then subscribed to the test images to identify the fields to remove. However, the user must highlight some semantically meaningful layout entities to build a good model. The above article proposed a method of information extraction based on the registration of parts of layout elements. The proposed model, though extremely simple, really requires human intervention.

And next is "A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents"**[9]**. This method uses knowledge of the logical structure of documents described in the high-level language. It verifies the localization of the logical entities extracted from the document content (characters, words), sentences, etc.) and physical entities removed from the document layout (markers, blocks, lines, etc.). Only a few applications use a top-down approach but can build a document comprehension system based on this approach. The bottom-up approach is time-consuming and much less elegant.

Recent research using all feature text and images to support the output work mainly depends on graph-based symbols due to the graphical market hoarding network (GCN). Successfully demonstrated a large company in mission non-data structure (Semi-supervised classification with complex network graph **[10]**). In general, GCN methods can be divided into spatial control methods and total control.

Convolution of the graph Our framework for obtaining a richer representation falls under spatial convolution, which broadly describes graph convolution operations by illustrating a process on node groups of neighbors. The first is that Graph attention networks **[11]** provide an attention-based architecture to perform node classification of graphically structured data. According to the self-attention strategy, the idea is to compute the hidden representations of each node in the chart by observing its neighboring nodes. The attention architecture has several interesting properties:

- The operation is efficient since it is parallelizable across node neighbor pairs.
- It can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors.
- The model is directly applicable to inductive learning problems, including tasks where the model has to generalize to completely unseen graphs.

The proposed approach on four challenging points is Cora, Citeseer, and Pubmed citation networks and an inductive protein-protein interaction dataset to achieve or compare modern results that highlight the potential of tissues. Shape-based attention when dealing with arbitrarily structured graphs.

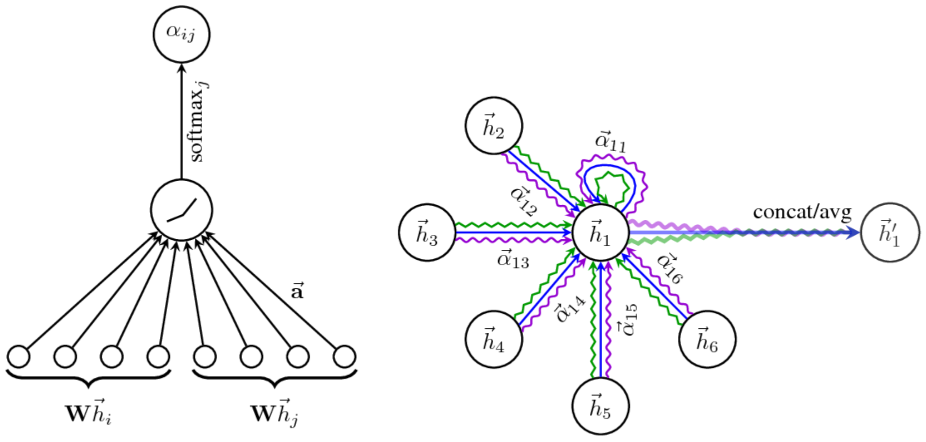


Figure 2.4. Left: The attention mechanism a(W~hi,W~hj) employed by our model, parametrized by a weight vector ~a R2F , applying a LeakyReLU activation. Right: An illustration of multihead attention (with K = 3 heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain ~h1 [11]

Next is Geometric deep learning on graphs and manifolds using mixture model CNNs**[12]**, which presents mixed model networks (MoNet). This common framework allows the design of deep structures that accumulate over non-Euclidean- like graphs and manifolds. The approach is to follow the general philosophy of spatial domain methods and build convolutional operations in pattern matching with local internal 'patches' on a graph or manifold. The main novelty is how the patch is extracted: whereas the previous approaches use fixed fixes. In particular, the above paper shows that the patch operators can be constructed as a function of the local graph or pseudo-manifold coordinates and study a family of functions represented as a mixture of particles Gauss multiplier. Such construction allows the formation of Geodesic CNN (GCNN) and Anisotropic CNN (ACNN) proposed earlier on manifolds or GCN and DCNN on the graph as specific cases of the approach. In the first problem layer, the task is to classify the image, which is considered the superpixel's adjacency graph. In the second type of problem, perform a vertex classification on a chart representing a grid of scientific papers. Finally, consider the problem of finding dense intrinsic correspondence between 3D shapes known as manifolds. In all the above issues, it is found that the above approach is always superior to the non-Euclidean deep learning methods previously proposed.

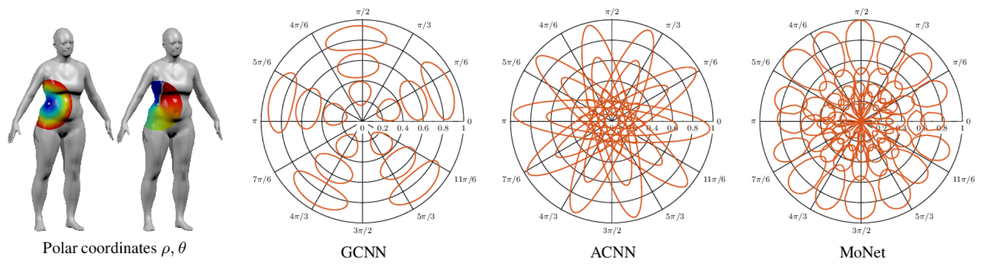


Figure 2.5. Left: intrinsic local polar coordinates , on a manifold around a point marked in white. Right: patch operator weighting functions wi(,) used in different generalizations of convolution on the manifold (hand-crafted in GCNN and ACNN and learned in MoNet). All kernels are L-normalized; red curves represent the 0.5 level set **[12]**.

However, spectral methods often determine graph convolution operations based on the semi-supervised classification with graph convolutional networks, which are not conducive to dynamic graph structures. The article N-ary Relation Extraction using Graph State LSTM**[13]** and Cross-Sentence N-ary Relation Extraction with Graph LSTMs**[14]** have proposed Graph LSTM, which allows creating a different number of dependencies at each cell. Relational extraction has been studied in journalism, web publishing, and biomedical as a central task in natural language processing. It helps detect clear facts, such as cause-and-effect, and predict a drug's effect on sudden cancer. Variables of a given gene in the biomedical field. While most existing jobs extract relationships in a sentence, the task of pulling a cross-sentence relationship is getting more and more attention. In 2017 Peng et al **[14]**, Extended relation extraction between sentences by detecting different relationships between several entity references (n-ary relations). They proposed a graphically structured LSTM to extract the n-ary association. The graph is constructed from input sentences with dependent edges, linking adjacent words and relationships between sentences so that syntactic and discourse information can be used to extract the relationship. To calculate the latency coding for each word, they first separated the input graph into two alternating directional charts (DAGs) by separating left-to-right edges from right-to-left edges. Two separate gated recurrent neural networks, which extend tree LSTM), were adopted for each single-directional DAG, respectively. Finally, for each word, the latent states of both directions are joined together as the final state. The bi-directional DAG LSTM model shows superior performance over some strong baselines, such as tree-structured LSTM, on the biomedical domain benchmark.

However, the two-dimensional DAG LSTM model suffers from some limitations. First, important information may be lost when converting a graph into two separate DAGs. Second, only ancestors and descendants can be incorporated for each term using LSTMs on both DAGs. Modeling an overall chart and learning its representation without splitting it into two DAGs is one possible solution to the problems described above. Due to the existence of cycles, the naive extension of tree LSTMs cannot serve this target. In 2017, convolutional network graph (GCN) and repeating network graph (GRN) had been proposed to represent graph structure for NLP tasks. Such methods encode a given graph by learning to decentralize the representations of neighboring nodes in the graph through their connected edges. While GCNs use CNNs to exchange information, GRNs perform periodic steps over and over to achieve this goal. To compare pretty with DAG LSTMs, in the lesson N-ary Relation Extraction using Graph State LSTM built a graph LSTM by extending Song et al. (2018), strictly follow the configuration of Peng et al**[14]** such as the source of the features and parameter settings. In particular, the total input chart is modeled as a single state, with the words in the chart being its sub-states. The state transitions are performed on the graph continuously, allowing the lengthy states to exchange information through the dependent and discourse edges. Each word increases its current state at each repetition step by getting information from the current state of adjacent wordsTherefore, with an increasing number of repetitive steps increasing, each dish will receive information from a larger context. Compared with the two-dimensional DAG LSTM, the above method has several advantages:

1. It preserves the original graph structure, ensuring that no data is lost.
2. By transferring information up and down from a parent, sibling information can be easily integrated.
3. Information exchange allows for more parallelization and can therefore be very efficient in computation.

In Cross-Sentence N-ary Relation Extraction with Graph LSTMs explores a common framework for n-ary cross-sentence relation extraction, based on graph extended short-term memory networks (LSTM diagrams). By applying the diagrammatic formula, the methodology framework in the article uses prior approaches based on a series of tree LSTM. It can incorporate a rich range of linguistic analysis to aid in the extraction of generation. The input relational classifier for the entity representations is learned from the entire text and can be easily extended to arbitrary relations. This approach also facilitates joint learning with friendly relationships where monitoring signals are more abundant.

Together extract entities and relationships through the design of a directed graph schema. Removing entities and relationships is a fundamental task of information mining (IE). There are close relationships between entities and relationships, as well as between relational labels in a sentence. For example, a Direct connection suggests Person and Location entities, and vice versa. The direct relationship (between “Cuong” and “Ha Noi”) can be inferred from the direct connection (between “Cuong” and “Thai Binh”) and Loc In (between “Thai Binh” and “Ha Noi ”).

The task has been traditionally resolved as a series of two separate sub-tasks: entity identification and relation extraction. This separation ignores the association between these two side quests. Joint exploitation of entities and relationships was able to integrate the information of entities and relationships and to achieve a better result in this task. The general patterns were studied using both statistical methods and neurological methods. Performance of statistical models mainly rely on complex feature engineering, and it is difficult to exploit global features.

In contrast, neural methods automatically learn non-local features by exploiting repeating neural networks to understand sentence-level representations and have given the most current results. However, most existing neural models extract separate entities and relationships, achieving joint learning only through parametric sharing but not general decoding. This results in a limitation that information between output entities and relationships cannot be fully exploited since no explicit features are used to model output-output dependencies. In 2017, Zheng et al. was the only exception, designing a new tagging scheme and converting the general extraction task into a tagging problem. In their standard model, the entities and relationships information is integrated into a unified tagging scheme and can be fully exploited. However, due to the conversion to tagging, this method only indirectly captures the output structure correspondences. It cannot define overlapping relationships (e.g., an entity can only have at most one connection).

The article "Joint Extraction of Entities and Relations Based on a Novel Graph Scheme" **[15]** outlined the solution of converting a joint task into an oriented graph by designing a new graph schema, solved by using The parse framework is based on conversion. Unlike traditional parsing tasks, the nodes in the output structure can have multiple or no heads. They propose a new transition system, a variant of the list-based arc-eager algorithm for non-projective tree parsing . By integrating each entity and their respective relational information step by step, the method can model fundamental dependencies not only between entities and relationships but also between relationships. A challenge to designing a forward-based parsing system is representing the parse states (i.e., the configuration) on which the transformation actions are classified. They borrowed the idea of neural parsing, designed a particular recursive neural network to model the underlying entity relationships and dependency relationships. Use BiLSTM to represent each sentence token to capture richer contextual information.

The paper "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling" **[16]** proposed a GCN version suitable for modeling the syntax dependency diagram to encode sentences to label semantic roles. Semantic role labels (SRL) can be informally described as the task of discovering who did what to whom. SRL is often considered an essential step in the standard NLP pipeline, providing information for downstream information mining and questioning tasks.

Semantic representations are closely related to syntax representations, even though the syntax-semantics interface is far from trivial. Due to these similarities and also because of the availability of accurate parsers for multiple languages, it seems natural to exploit syntax information when predicting semantics. Although historically, most of the SRL approaches to rely on the last generation syntax of the SRL model put the syntax aside in favor of neural chain models, namely LSTM, and beyond. The authors of the above paper believe that one reason for this radical choice is the lack of efficient and straightforward ways to incorporate syntactic information into sequential (specific) neural networks. It is on the word level). In this article, they have proposed a workaround for this limitation.

Specifically, they are based on graph convolutional networks (GCNs), a kind of multi-layered neural network operating on recent graphs. For each node in the diagram (in our case, one word in a sentence), the GCN encodes the relevant information about its neighborhood as a true-valued feature vector. The GCN has mainly been studied in the context of undirected non-label graphs. They introduced the GCN version for syntax-dependent structure modeling and general application to labeled directional charts

And in the article, A Lexicon-Based Graph Neural Network for Chinese NER**[17]**, Tao Gui and his colleagues proposed a vocabulary-based GCN with global semantics to avoid word ambiguity. The above post introduced a vocabulary-based graph neural network (LGN) that achieves the Chinese NER as a node classification task. The proposed model breaks down the serialization processing structure of RNNs due to better interaction between letters and words through careful connections. Vocabulary knowledge connects relevant characters to capture local layouts. Meanwhile, a global forward button is designed to capture long-range dependency and high-end features. LGN follows a neighborhood composite diagram in which the node representation is computed by recursively summing its incoming edges and a global forwarding node. Due to the aggregation of repetitions, the model may use global context information to compare ambiguous words for better prediction repeatedly.

2.2. Related methods

However, their methods do not incorporate visual features into the model. The works most related to our method are "GraphIE: A graph-based framework for information extraction"**[18]** and "Graph convolution for multimodal information extraction from visually rich documents"**[19]** using graph modules to capture geographic objects.

2.2.1. GraphIE: A graph-based framework for information extraction

The GraphIE method is a prediction system that learns the relationships between local and non-local dependencies in the input space automatically. The algorithm works on a graph, with nodes representing textual units (words or sentences) and edges describing their relationships. A recurrent neural network sequentially encodes local contextual representations in the model, and then the graph module uses graph convolutions to iteratively propagate information between neighboring nodes (Kipf và Welling, 2016) **[43]**. Finally, the learned representations are projected back to a recurrent decoder to enable word-level tagging.

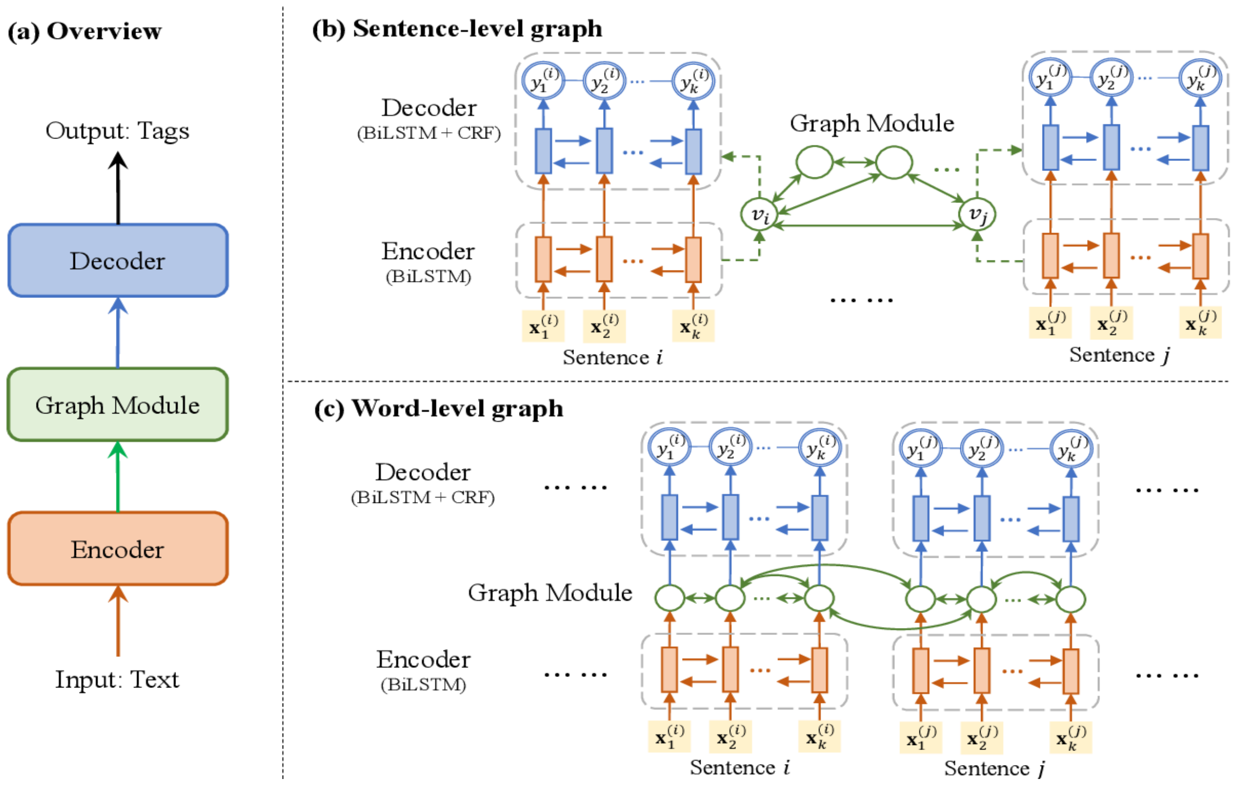


Figure 2.6. GraphIE framework: (a) an overview of the framework; (b) architecture for sentence-level graph, where each sentence is encoded to a node vector and fed into the graph module, and the output of the graph module is used as the initial state of the decoder; (c) architecture for word-level graph, where the hidden state for each word of the encoder is taken as the input node vector of the graph module, and then the output is fed into the decoder [18]

2.2.2. Graph convolution for multimodal information extraction from visually rich documents

The method combines textual and visual information presented in Visually rich documents (VRDs). Graph convolution produces graph embeddings that summarize the meaning of a text segment in a document, which is then combined with text embeddings for entity extraction using a standard BiLSTM-CRF model.

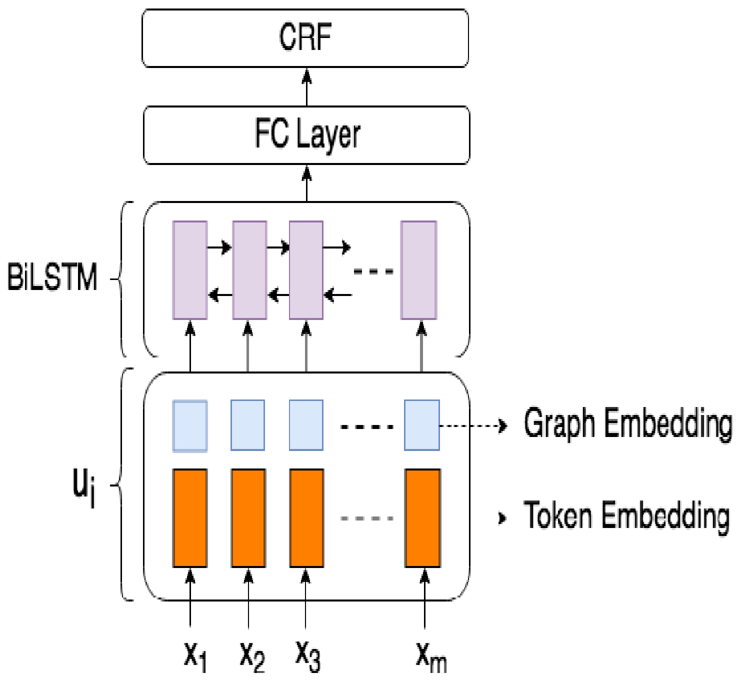


Figure 2.7. BiLSTM-CRF with graph embeddings. [19]

Method and multimodal to extract but still differ from us in some respects. Our process, however, incorporates graph learning into the framework, which can filter useless and powerful buttons to capture complex layout structures.

CHAPTER 3. METHODOLOGY

3.1. Convolutional neural networks(CNN or ConvNet)

Convolutional neural networks(CNN or ConvNet) are a network architecture used to directly retrieve valuable features from the initial data, removing the need for time-consuming and expensive manual extraction. It is beneficial for locating subjects, faces, and scenes. CNN is often used to describe non-visual data including audio, time series, and signal data**[20]**.

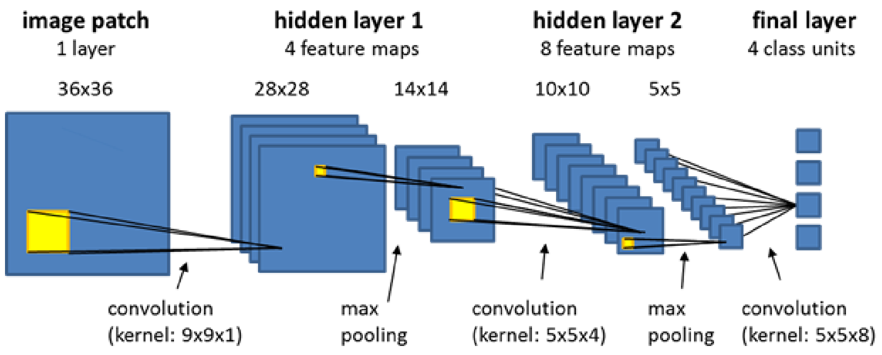


Figure 3.1. A CNN sequence to classify handwritten digits**[20]**

3.1.1. What Makes ConvNets So Useful?

The use of ConvNets for deep learning is every day for three reasons:

- ConvNets remove the need for manual feature extraction since CNN knows the features directly.
- ConvNets generate largely accurate identification records
- ConvNets may retrain to perform new recognition activities, allowing you to bring on previously qualified networks.

3.1.2. How ConvNets Work?

A ConvNet, like most neural networks, is made up of an input layer, an output layer, and several hidden layers in between.

These layers perform operations on the data to learn data-specific features. The typical popular layers in ConvNet are convolution, activation or ReLU, and pooling.

- **Convolution** processes the input images through a series of convolutional filters, each of which stimulates different aspects of the images.
- **Rectified linear unit (ReLU)** maps negative values to zero while preserving positive values, so its planning is effective and quicker. It is often known as activation, and only the activated features that have been enabled are taken on to the next layer. Other activation functions are used regularly as sigmoid, tank, leaky relu.
- **Pooling** decreases the number of parameters that the network must learn by doing nonlinear downsampling on the output.

These operations are replicated several times over tens or hundreds of layers, with each layer learning to recognize various features.

3.2. Graph Convolutional Networks

3.2.1. Basic knowledge of graph

Graph

A graph is represented as G = (V, E) with:

- G : graph
- V: a set of nodes of the graph (vertices / nodes)
- E: the set of edges connecting the nodes of the graph (edge).
- $e_{ij} = (v_i, v_j) \in E$ is the representation of the edges $e \in E$ connecting from node v_i to node v_j of the graph

Adjacency Matrix

Adjacency matrix A, is a square matrix of size nxn (where n is the total number of nodes in the graph).

- $A_{ij} = 1\, if\, e_{ij} \in E$
- $A_{ij} = 0\, if\, e_{ij} \notin E$
- The weights of the sides of the graph are expressed by the Adjacency matrix (A), also known as a weighted-matrix.

Degree matrix

Degree matrix D, is a diagonal matrix nxn, containing the degree information of each vertex, with

- $D_{ii} = \sum_{i=1}^{n} A_{ij}$
- Note that with a direct matrix, the order of each node counts only edges connected to that node

Identity matrix

Identity matrix I, is a diagonal matrix nxn, with prices on the main diagonal = 1, the remainder = 0

- $I_{ij} = 1\, if\ i\ == \ j,\ else\, I_{ij} = 0$

Self-loop: a button with edge connecting itself to itself

Laplacian matrix: L = D - A

Frobenius norm:

The Frobenius norm is the matrix norm of a $m\ x\ n$matrix A that is defined as the mixed root for the sum of absolute squares of its elements, sometimes known as a euclidean norm, (unfortunately also used for the vector L^2-norm).

$$||A||_F = \sqrt{\Sigma_{i=1}^{m}\Sigma_{i=1}^{n}|a_{ij}|^2} \quad \textbf{(1)}$$

Directed graph and undirected graph:

- Undirected graph or undirected matrix, when the edge between vertices i and j is the same, or $e_{ij} = \ e_{ji}$
- Direct graph or directional matrix, has a defined dimension from node $v_i$to node v_j, and there is a linkage next to e_{ij}.

3.2.2. What are Graph Convolutional Networks ?

Related operations are performed by GCNs, in which the model learns the features by evaluating adjacent nodes. The main distinction between CNNs and GNNs is that CNNs are designed to work with regular (Euclidean) organized data, while GNNs are a simplified variant of CNNs in which the number of nodes links varies and the nodes are not in any particular order (irregular on non-Euclidean structured data)**[21]**.

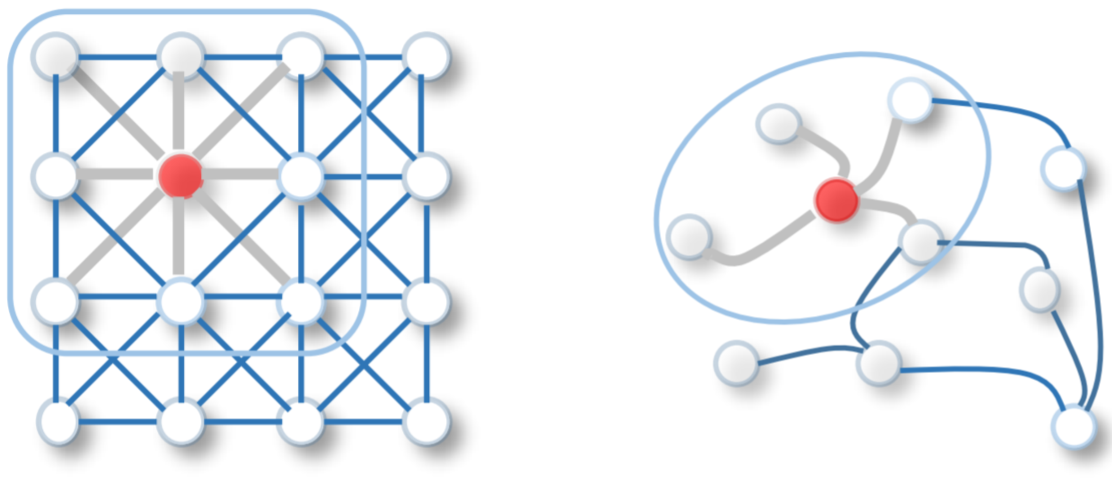


Figure 3.2. 2D Convolution (left) and Graph Convolution(right)**[21]**

GCNs can be classified into two types of algorithms:

- Spatial Graph Convolutional Networks
- Spectral Graph Convolutional Networks.

Fast Approximate Spectral Graph Convolutional Networks

The initial inspiration for Spectral GCN came from signal/wave propagation. Knowledge propagation in Spectral GCN can be thought of as signal propagation around the nodes.

To apply this method of knowledge transmission, spectral GCNs use the Eigen-decomposition of the graph Laplacian matrix. Stated, the Eigen-decomposition assists in the comprehension of graph form and hence the recognition of graph nodes.

In Neural Networks, we execute the following equation to propagate the features representation to the next layer (forward pass):

$$H^{[i+1]} = \sigma(W^{[i]} H^{[i]} + b^{[i]}) \qquad (2)$$

In GCN to the node features (or so-called input features) in this process, we can use the Adjacency Matrix (A) in the forward propagation equation. The forward pass equation will then be:

$$H^{[i+1]} = \sigma(W^{[i]} H^{[i]} A^{*}) \qquad (3)$$

- H^{i} represents the output of layer i + 1, each layer H^{i} corresponds to a matrix of size N x F^{i}. With F^{i} shows the number of output features of each node at layer H^{i}.
- σ is an activation function: Softmax, ReLu, Leaky Relu,...

- $b^{[i]}$ is bias at layer i
- H^{i} = X, where X is the initialized weight, node feature of each node
- $W^{i}$is the weight matrix corresponding to the ith layer
- N is the total number of nodes
- A (Adjacency Matrix) matrix that describes the edges or relations between the nodes in the forward propagation equation is known as an adjacency matrix. The addition of A to the forward pass equation helps the model to learn function representations dependent on node connectivity.
- A^{*} is the normalized version of A.

Why do we have to normalize adjacency matrix A?

Initializing the Graph G

As seen below, we have

- An undirected graph with 6 nodes
- Adjacency matrix A.
- Name of the graph is G.
- Number of edges is 7.
- Average degree: 2.3333
- Graph node: [(0, {‘name’:0}), (1, {‘name’:1}), (2, {‘name’:2}), (3, {‘name’:3}), (4, {‘name’:4}), (5, {‘name’:5})]

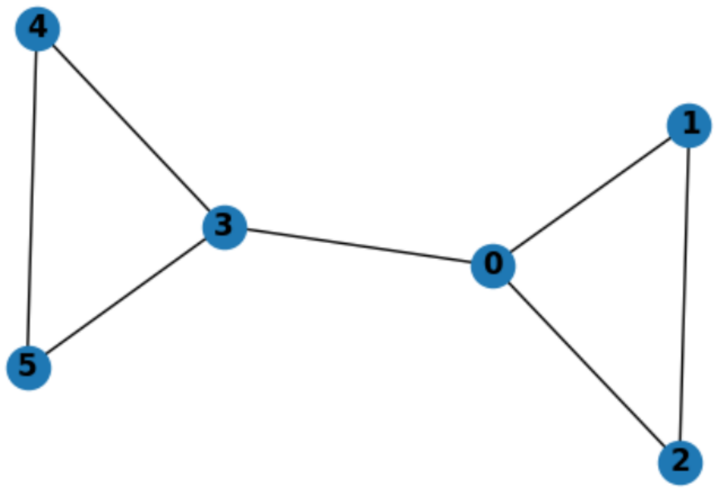


Figure 3.3.a. Graph G visualization

Adjacency Matrix (A) Insertion into Forward Pass Equation

The Adjacency Matrix (A) and Node Features Matrix (X) of graph G are obtained next.

- Shape of A: (6, 6)
- Shape of X: (6, 1)
- Adjacency Matrix (A):

```
[[0. 1. 1. 1. 0. 0.],
 [1. 0. 1. 0. 0. 0.],
 [1. 1. 0. 0. 0. 0.],
 [1. 0. 0. 0. 1. 1.],
 [0. 0. 0. 1. 0. 1.],
 [0. 0. 0. 1. 1. 0.]]
```

- Node Features Matrix (X):

```
[[0],
 [1],
 [2],
 [3],
 [4],
 [5]]
```

Inserting Self-Loops (A_hat) and calculate AX that represents the sum of neighboring nodes features

- Edges of G with self-loops:

[(0, 1), (0, 2), (0, 3), (0, 0), (1, 2), (1, 1), (2, 2), (3, 4), (3, 5), (3, 3), (4, 5), (4, 4), (5, 5)]

- Adjacency Matrix of added self-loops G (A_hat):

```
[[1. 1. 1. 1. 0. 0.],
 [1. 1. 1. 0. 0. 0.],
 [1. 1. 1. 0. 0. 0.],
 [1. 0. 0. 1. 1. 1.],
 [0. 0. 0. 1. 1. 1.],
 [0. 0. 0. 1. 1. 1.]]
```

- AX:

```
[[ 6.]
 [ 3.]
 [ 3.]
 [12.]
 [12.]
 [12.]]
```

You would be able to see another issue now. AX's components are not normalized. In order for the model to converge, we must normalize the features, much as we might for every other Neural Networks activity. This **avoids numerical instabilities and vanishing/exploding gradients**. We normalize our data in GCNs by computing the Degree Matrix (D) and conducting a dot product function with AX on the opposite of D.

$$normalized\ features = D^{-1}AX \quad (4)$$

- Degree Matrix of added self-loops G (D):

```
[(0, 5), (1, 4), (2, 4), (3, 5), (4, 4), (5, 4)]
```

- Degree Matrix of added self-loops G as numpy array (D):

```
[[5 0 0 0 0 0],
 [0 4 0 0 0 0],
 [0 0 4 0 0 0],
 [0 0 0 5 0 0],
 [0 0 0 0 4 0],
 [0 0 0 0 0 4]]
```

- Inverse of D:

```
[[0.2  0.   0.   0.   0.   0.  ],
 [0.   0.25 0.   0.   0.   0.  ],
 [0.   0.   0.25 0.   0.   0.  ],
 [0.   0.   0.   0.2  0.   0.  ],
 [0.   0.   0.   0.   0.25 0.  ],
 [0.   0.   0.   0.   0.   0.25]]
```

- DAX:

```
[[1.2 ],
 [0.75],
 [0.75],
 [2.4 ],
 [3.  ],
 [3.  ]]
```

If we compare DAX with AX, we will notice that:

```
  AX:          DAX:
[[ 6.],      [[1.2  ],
 [ 3.],       [0.75],
 [ 3.],       [0.75],
 [12.],       [2.4  ],
 [12.],       [3.    ],
 [12.]]       [3.    ]]
```

The problem is based on a clustering graph, evaluating an important information node based on the number of edges connected to it **[22]**. Looking at Figure 3.5, we see that node 3 has the number of connected edges 3, node 4,5 has the number of connected edges 2. Also, before normalization, AX, weights at node 3 and node 4,5 Equality is 12. Furthermore, after normalization, $D^{-1}AX$, node 3 (weight 2.4) has a higher weight than node 4,5 (weight 3). The fact that $D^{-1}AX$is more accurate than $D^{-1}AX$, or the lower the degree of a node, the stronger a node belongs to a particular group or cluster.

***Renormalization trick* [23]**:

Kipf and Welling state in the paper that symmetrical normalization makes dynamics more interesting, and therefore modifies the standardization equation from:

$$normalized\ term\ =\ D^{-1}A$$
$$to$$
$$normalized\ term\ =\ D^{-1/2}AD^{-1/2}$$

```
   DADX:
[[1.27082039],
 [0.75      ],
 [0.75      ],
 [2.61246118],
 [2.92082039],
 [2.92082039]]
```

3.2.3. Graph Convolutional Networks architecture

In hidden layers, GCN performs the following layer-wise propagation,

$$H^{i+1}\ =\ \sigma(D^{-1/2}AD^{-1/2}H^{i}W^{i})\ \ (5)$$

GCN defines the final perceptron for semi-supervised node classification as follows:

$$Z\ =\ softmax(D^{-1/2}AD^{-1/2}H^{K}W^{K})\ \ (6)$$

The number of classes is represented by $W^{K}\ \in\ R^{d_{K}x\,c}$ and c. The Z ∈ $Z\ \in\ R^{n\,x\,c}$ final output denotes the label prediction for each X data row, whereby each Z_{i} rows denotes the i-th node label prediction.

The optimal weight matrices W = {$W^{(0)}$, $W^{(1)}$, · · · $W^{(K)}$} are trained by minimizing the following cross-entropy loss function over all the labeled nodes L, i.e.,

$$L_{Semi-GCN}\ =\ -\ \Sigma_{i\in L}\Sigma_{j=1}^{c}Y_{ij}lnZ_{ij}\qquad (7)$$

Where the labeled node set is indicated by L and $Y_{i},\ i\ \in\ L$ denotes the respective labeling indication of the labeled i-th node.

3.3. Graph Learning-Convolutional Networks

A new Graph Learning-Convolution Network (GLCN) **[24]** will learn an adaptive (or optimal) graph representation for GCN learning by *combining* **graph learning** and **graph convolution** in a single network architecture.

3.3.1. Graph learning architecture

- Given an input X = $(x_1, x_2 \ldots x_n) \in R^{pxn}$
- S_{ij}= g(x_i , x_j) a nonnegative function that represents the pairwise relationship between data x_i and x_j
- $a = (a_1, a_2 \ldots a_p)$ T $\in R^{px1}$is a weight vector

$$S_{ij} = g(x_i, x_j) = \frac{exp(ReLU(a^T|x_i - x_j|))}{\Sigma_{j=1}^{n} exp(ReLU(a^T|x_i - x_j|))} \quad (8)$$

The role of the above softmax operation on each row of S is to guarantee that the learned graph S can satisfy the following property,

$$\Sigma_{j=1}^{n} S_{ij} = 1, S_{ij} \geq 0$$

We optimize the optimal weight vector a by minimizing the following loss function,

$$L_{GL} = \Sigma_{i,j=1}^{n} ||x_i - x_j||_2^2 S_{ij} + \gamma||S||_F^2, (9)$$

3.3.2. Graph Learning-Convolutional Networks architecture

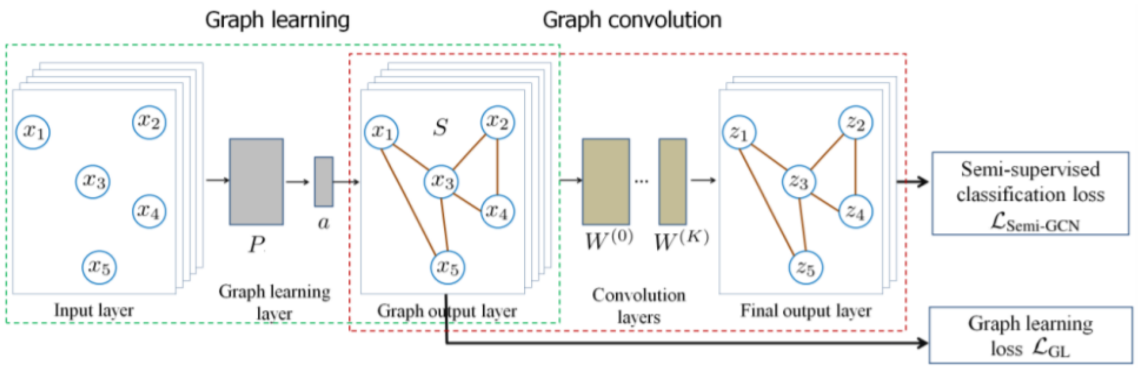


Figure 3.3.b. Architecture of the proposed GLCN network for semi-supervised learning.**[24]**

From $Eq. 7$ and $Eq. 9$, minimizing the following loss function as:

$$L_{Semi-GCN} = L_{Semi-GCN} + \lambda L_{GL}$$

3.4. Bidirectional long short term memory(Bi-LSTM) and Conditional random field(CRF)

3.4.1. What is Bidirectional long short term memory(Bi-LSTM)?

The BI-LSTM is based on the model of recurrent neural networks (RNN) and is developed from LSTM.

Recurrent Neural Networks

A recurrent neural network(RNN) keeps a memory centered on past data, allowing the model to forecast current performance based on long-distance features. RNNs are used for sequential prediction activities, such as speech recognition**[25,26]** and are capable of predicting the next term in a sense**[27]**. RNN has many different types such as: one to one, one to many, many to one, many to many.

Figure 5.1a is RNN structure **[28]** with:

- x_t is an input of step t, as a one-hot vector.
- s_t is hidden memory in step t. It is computed using the front hidden state s_{t-1} and input x_t at that step. $s_t = f(Ux_t + Ws_{t-1})$. The f function is a nonlinear function as *Tang Hyperbolic (tank)* or *Rectified Linear Units (ReLu)*. The initialization value is regularly set to 0.
- o_t is an output of step t.

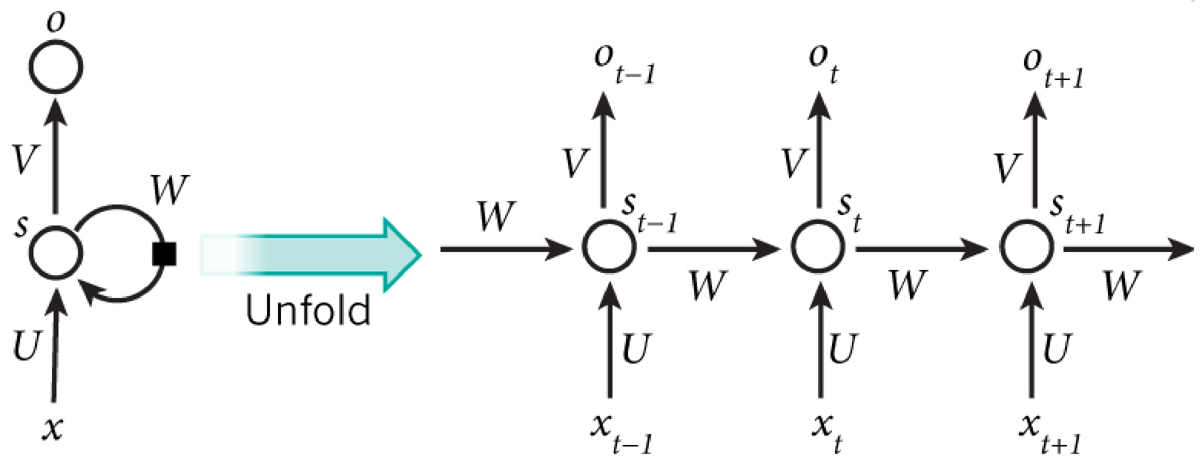


Figure 3.4.a. RNN structure **[28]**.

While RNN has made significant advances in Machine Translation or Language Modeling**[29,30]**, Speech Recognition**[31]**, Generating Text**[32]**, and Generating Image Descriptions**[33]** . However, the algorithm can only estimate correctly for short sequences. For example 1, if we have the phrase "That student is going to school", we can deduce that the network is "school" by reading "That student is going to". Since the gap between the acquired knowledge used for prediction is short in this case, the RNN can be fully known. But in many situations, we are forced to use more context for inference. For the second example, predict the last word in the paragraph: “I grew up in Vietnam… I speak fluent Vietnamese.”. Obviously, the information near ("I speak fluently") only allows us to know that behind it will be the name of a language. It is impossible to know what language it is. To know what language it is, we need to have the context "I grew up in Vietnam" again to be able to deduce. Clearly, the information gap might be quite far now. As a result, RNN Model is difficult to forecast if the sequence is long, and Long Short-Term Memory(LSTM) has resolved this**[34]**.

Long Short-Term Memory (LSTM)

The cell status is crucial to the LSTM[35]. The cell status is similar to that of a conveyor belt. It passes across all network nodes (links) with just a tiny linear interaction. Therefore, information can be easily transmitted without fear of change.

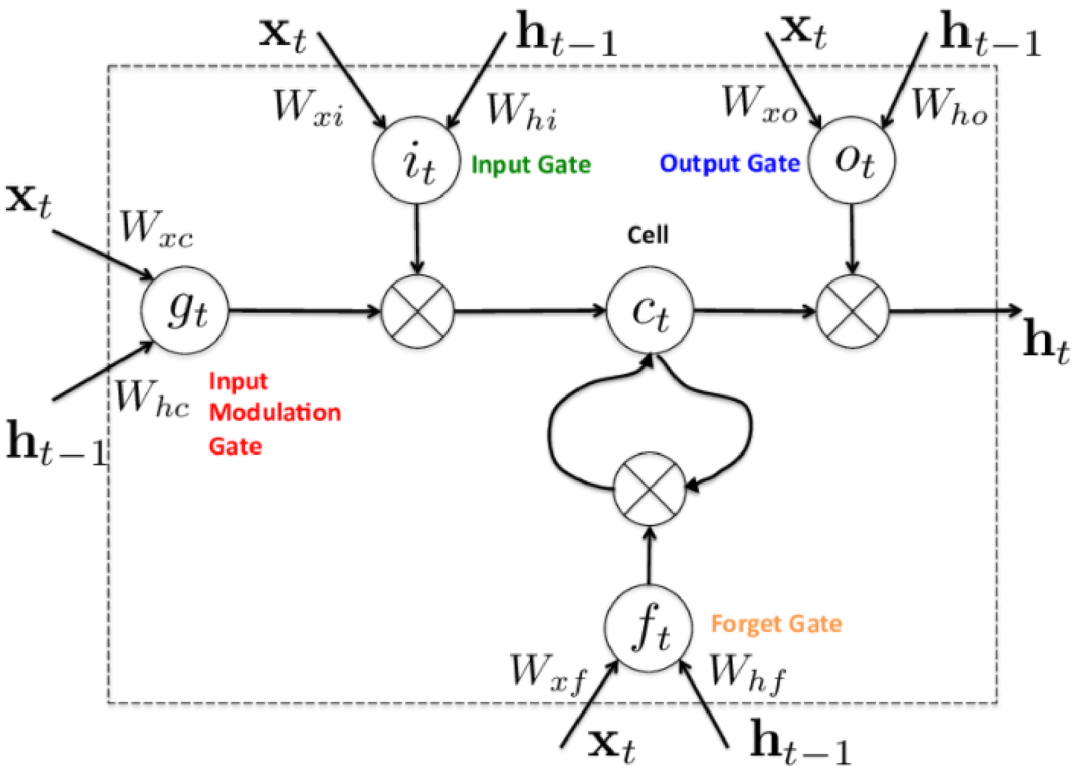


Figure 3.4.b. It is a unit structure of LSTM, including 4 gates: input modulation gate, input gate, forget gate and output gate.

- **Forget gate:** This port determines what information in recent memory is kept and what information is left behind. Input information is given to the sigmoid.
- **Input gate:** This port is used to update the memory with new information. Here appeared 2 sigmoid and fishy functions.
- **Output gate:** This port determines what the output of the current word is. It gets information from two sources: current cell status and input. The modified cell state goes through the tanh function, and the current input is passed through the sigmoid function. From here, we combine the two results above to get the output. Note that both the output and the cell state are included in the next step.

Bidirectional long short-term memory (Bi-LSTM) architecture

The Bi-LSTM system will run the data from one past to another and from one future to the next in two ways. What differs from unidirectional is that it conserves the future using the LSTM, which runs backward and combines the two hidden states. It can keep information from the past as well as from the future at any time.

Let us assume we try to predict the next word in a phrase. What a single LSTM would see on a high stage is "The boys went to ….". With bidirectional LSTM, you will see details on the path further and can attempt to guess the next term in this sense. As in NLP, often recognizing a term requires one to look not just at the previous word, but also at the next word, as in this example: "Mom said, Donald Duck stands by the field," and "Mom said, Donald Trump has become the president" Here for the word "Donald," we cannot just say whether the next word is going to be "Duck" or "Trump." It will depend on the context of the sentence.

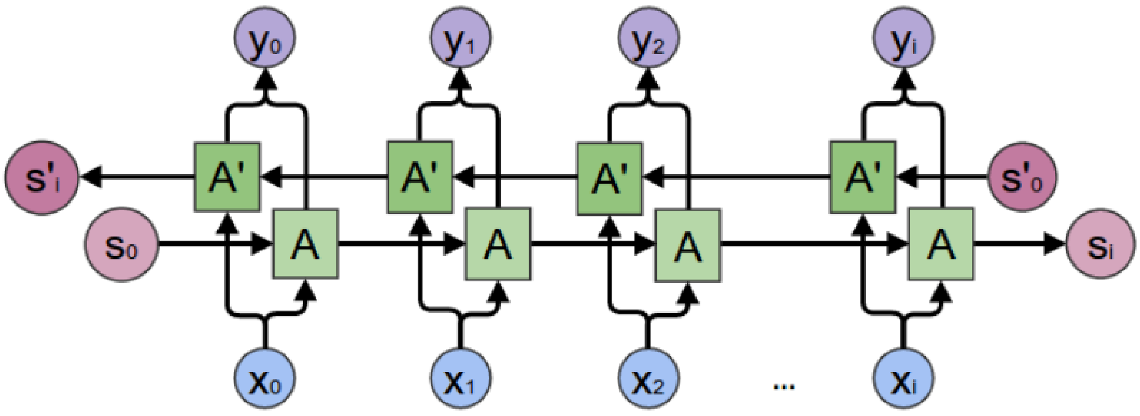


Figure 3.4.c. Bi-LSTM structure

3.4.2. What is Conditional random field?

A conditional random field**[36]** is an algorithm that graphically classifies data, and its ability to model several variables is dependent on each other. CRF derives from naive Bayes used to classify discrete data of graphs. Even when naive Bayes has good classification accuracy, its probability estimates are typically low. Thus, the solving path for a discrete graph is given by the joint distribution p (y, x) of an HMM with conditional probability p (y | x), also known as a linear-chain conditional random field.

Let Y,X be random vectors, $f_k(y_t, y_{t-1}, x_t)$ be a set of real-valued feature functions, $\theta = \{\lambda_k\}$ be a set of parameters of CRF . The linear-chain CRF is a distribution p(y|x) that takes from

$$p(y|x) = \frac{1}{Z(x)} exp\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\} \qquad (10)$$

Z(x) is an instance-specific normalization function

$$Z(x) = \sum_{y} exp(\lambda_k f_k(y_t, y_{t-1}, x_t) \qquad (11)$$

The penalized maximum likelihood estimator is a standard technique for parameter estimation. When we are only working with the Gaussian assumptions, the log-likelihood that uses a conditional distribution is suitable:

$$L(\theta) = \sum_{i=1}^{N} log\, p(y^i|x^i) \qquad (12)$$

where $x^i = \{x_1^i, x_2^i, x_3^i, ..., x_T^i\}$is a sequence of inputs, $y^i = \{y_1^i, y_2^i, y_3^i, ..., y_T^i\}$is a sequence of the desired predictions.

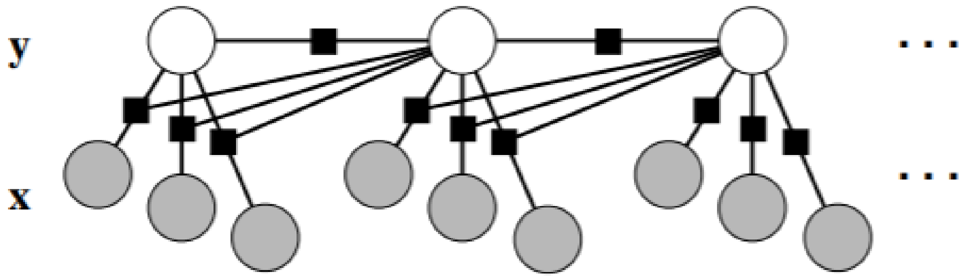


Figure 3.4.d. Graphical model of a linear-chain CRF in which the transition score depends on the current observation[36].

3.5. Transformer

The original transformer**[37]** is an encoder-decoder dependent, the neural network described in the paper *Attention is All You Need (2017)*. The key feature of the transformer is the so-called attention. That is a way to determine the relative importance of phrases in a single sentence or of the words that are most probable.

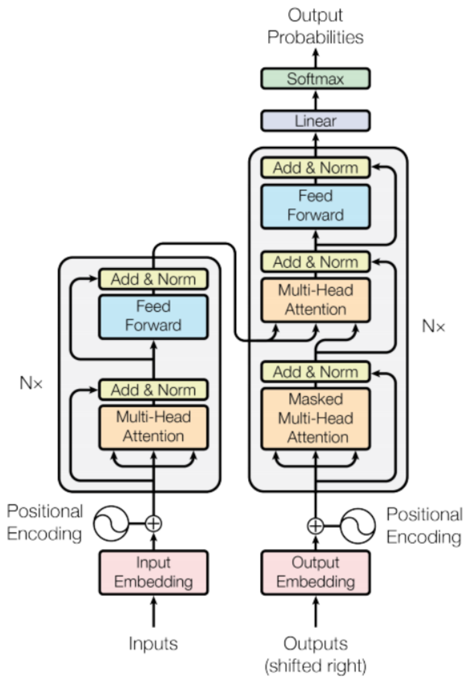


Figure 3.5.a. The Transformer - model architecture [37].

There are three elements worth analyzing:

- **The multi-head attention (orange)**: Adapted from the Bahdanau 2014 **[38]** definition, with algorithm optimization. According to the principle, a conditioning signal or query is applied to a set of key-value pairs — the query and key interact in any way, resulting in some normalized weights. These weights are then added to the value to produce a weighted number.
- **The position-wise feed-forward networks (light blue):** Position-wise feed-forward networks can think about in two respects. Either they are a two-layer, fully connected ReLU network applied to every location. Alternatively, there may be two 1-kernel-size convolutions applied throughout location space (which I prefer): Conv → ReLU → Conv.
- **The positional-encoding:** The authors decide to use fixed sinusoids of various frequencies that are applied directly to the input embeddings to inject location information. It's similar to a Fourier transform. Learned positional encodings often operate, although the authors expect that this would assist with longer sequence generalization. In any case, it's a smart solution since it allows for simple modeling of relative locations using linear functions.

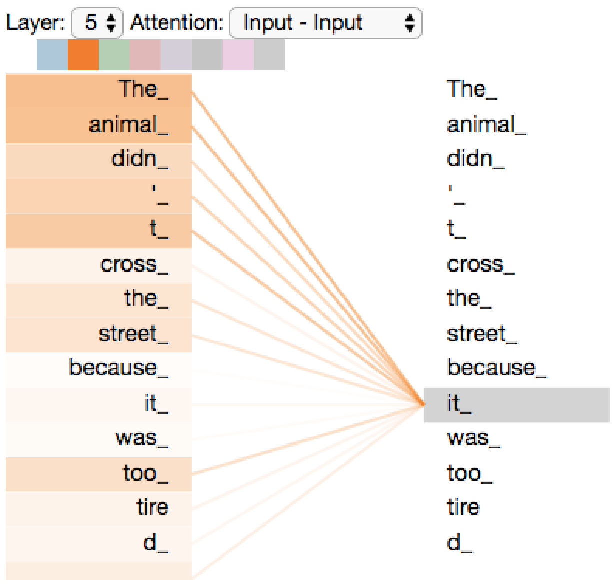

Layer: 5 Attention: Input - Input

Figure 3.5.b. Main idea of self-attention in transformer-translation structure

3.6. Proposal Model

Figure 3.6. includes:

- **Encoder**: This module encodes text segments to obtain text embeddings and image segments to *obtain image embeddings* using *Transformer and CNN*. Text and image segments represent textual and morphological information, respectively. The two types of embeddings are then combined to form a new local representation X, which is used as node input to the Graph Module.

- **Graph Module**: Through *improved graph learning convolutional operation*, this module can *capture the latent relationship between nodes* and obtain *richer graph embeddings* representation of nodes. Meanwhile, bounding boxes containing the document's layout context are modeled into the graph embeddings so that graph modules can *obtain non-local* and *non-sequential features*.
- **Decoder**: After obtaining the graph embeddings of the document, this module *uses BiLSTM and CRF* to perform sequence tagging on the *union non-local* sentence at *the character level*. By taking into account the layout information and the global information of the document, our model transforms key information extraction tasks into a sequence tagging problem.

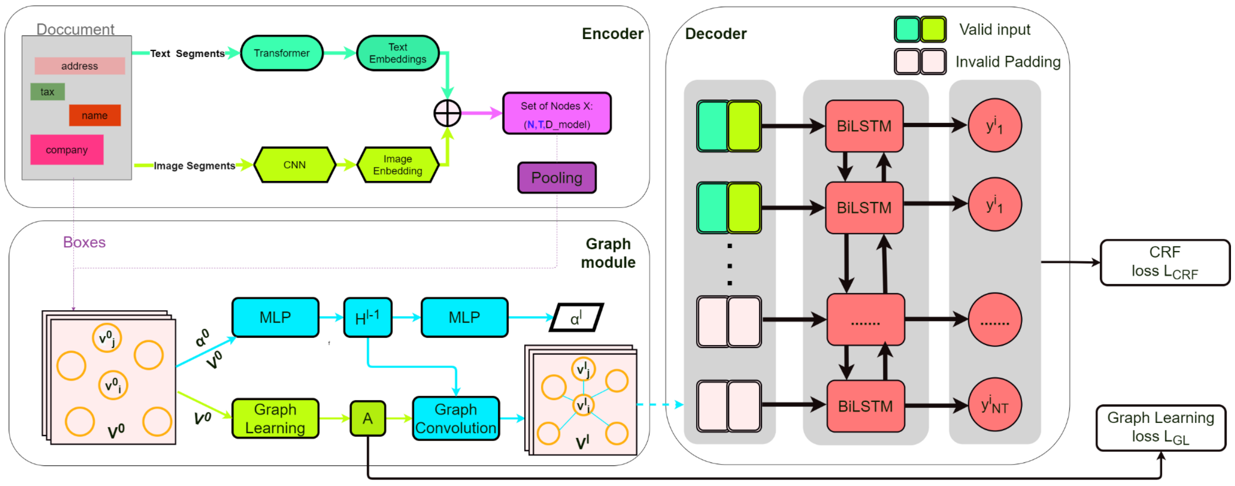


Figure 3.6. Overview of proposal model. V^l donates *node embedding* in the l-th graph convolution layer. α^l and H^l represents *relation embedding* and *hidden features* between the node v_i and v_j in the l-th graph convolution layer, respectively. A is *soft adjacent matrix*. N, T, and D M odel denotes the number of sentences segments, the max-length of sentence and the dimension of the model respectively. ⊕ denotes element-wise addition.

3.6.1. Notation

Given a document D that contains *N sentences/text segments*, the representation of it is denoted by S = $\{s_1, \ldots, s_N\}$, where s_i *denotes the character set for the i-th sentence/text segments. At position i*, we refer to s_i^{is} *as image segments* and s_i^{bb} *as a bounding box*. We use the IOB (Inside, Outside, Begin) tagging scheme to label each character as $y_i = (y_1^{(i)}, \ldots, y_T^{(i)})$ sequentially for each sentence $s_i = (c_1^{(i)}, \ldots, c_T^{(i)})$, where T is the length of sentence s_i.

A document D's accessory graph is denoted by $G = (V, R, E)$, 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 ⊂ V × R × V is the edge set and each edge eij = (v_i, α_{ij}, v_j) ∈ E represents that the relation α_{ij} ∈ R exists from node v_i to v_j.

3.6.2. Encoder

The encoder module, depicted in Figure 3.6., is located in the top-left corner of the diagram and contains two branches. Unlike existing key information works **[18], [19]** only uses text segments or bounding boxes. One of the most significant contributions of this paper is that we use picture fragments with morphology detail to refine text representations, which can then be used to assist key knowledge extraction tasks.

To capture local text contexts, one Encoder branch employs a transformer encoder**[37]**. Given a sentence $s_i = (c_1^{(i)}, \ldots, c_T^{(i)})$, text embeddings of sentence s_i is defined as follows

$$te_{1:T}^{(i)} = TransformerEncoder(c_{1:T}^{(i)};\ \Theta_{tenc}),\ (13)$$

where $c_{1:T}^{(i)} = [c_1^{(i)}, \ldots, c_T^{(i)}]^T \in R^{T\,x\,d_{model}}$ denotes the input sequence, $c_t^{(i)} \in R^{d_{model}}$ represents a token embedding (e.g., Word2Vec) of each character $c_T^{(i)}$, d_{model} is the dimension of the model, $te_{1:T}^{(i)} = [te_1^{(i)}, \ldots, te_T^{(i)}]^T \in R^{T\,x\,d_{model}}$ denotes the output sequence, $te_1^{(i)} \in R^{d_{model}}$ represents the encoder output of Transformer for the i-th character $c_T^{(i)}$, and Transformer's encoder parameters are described by Θ_{tenc}. Each sentence is encoded separately, and we can obtain a document D text embeddings, which we define as

$$\mathbf{TE} = [te_{1:T}^{(1)}; \ldots ; te_{1:T}^{(N)}] \in R^{N\,x\,T\,x\,d_{model}}\ (14)$$

By using CNN to record morphology, another encoder branch generates image injection. Given a image segment s_i^{is}, image embeddings is defined as follows

$$ie^{(i)} = \mathbf{CNN}(s_i^{is};\ \Theta_{CNN}),\ (15)$$

where s_i^{is} is $\in R^{H'\,x\,W'\,x\,3}$ denotes the vector of input image segment, H' and W' represent the height and width of image segment s_i^{is} respectively, $ie^{(i)} \in R^{H\,x\,W\,x\,d_{model}}$ represents the output of CNN for the i-th image segment s_i^{is}, and Θ_{CNN} represents the CNN parameters. We use ResNet **[39]** to execute the CNN and resize the image under the condition $H \times W = T$ then encode each image segments individually and we can get a document D image embeddings, defining it as

$$\mathbf{IE} = [ie^{(i)}; \ldots ; ie^{(N)}] \in R^{N\,x\,T\,x\,d_{model}}. \quad (16)$$

Finally, by an additional elemental operation, we combine text embeddings TE and image embedding IE and then generate the fusion embedding X from document D, which can be expressed as

$$\mathbf{X} = \mathbf{TE} + \mathbf{IE}\ , \quad (17)$$

where X $\in R^{N\,x\,T\,x\,d_{model}}$represent a set of nodes of graph ant X will be used as input X_0 of Graph Module followed by pooling operation and $X_0 \in R^{N\,x\,T\,x\,d_{model}}$.

3.6.3. Graph Module

Existing main information functions**[18]** and **[19]** are used to model global design context and non-sequential information using graph neural networks. Prior knowledge is needed for the predefined task-based edge type and the graph's adjacency matrix. **[18]** define the edge to wide, horizontally, or vertically connected nodes/text and identify four adjacent matrix forms (left-to-right, right-to-left, up-to-down, and down-toup). But this method cannot make full use of all graph nodes and excavate latent connected nodes that are far apart in the document. Although **[19]** uses a fully connected graph that every node/text segment is connected, this operation leads to graph aggregate useless and redundancy node information.

We are incorporating better graphical learning networks inspired by**[10]** in the current graph architecture, as seen in the bottom left of Figure 3.6., so as to obtain a soft adjacent matrix A for the graphical background to be used for downstream tasks.

Graph Learning

Given an input, V = $[v_1, \ldots, v_N]^T \in R^{N\,x\,d_{model}}$ of graph nodes, where $v_i \in R^{d_{model}}$ is the i-th node of the graph and the initial value of V is equal to X_0, Graph Module generates a soft adjacent matrix A, which represents, first of all by graphic learning, the weight of the relationship between two nodes and extracts H for each v_i node by means of a multi-layered perceptions network (MLP) just as **[19]** for Vinput and the corresponding α integrations. Then we operate the H feature graphics, spread information between nodes, and add that information to a new $V^{'}$ feature. Mathematically, we learn a soft adjacent matrix A using a single-layer neural work as

$$\begin{cases} A_i = softmax(e_i),\ i = 1,\ldots,N,\ j = 1,\ldots,N, \\ e_{ij} = LeakRelu(w_i^T|v_i - v_j|)) \end{cases} \quad (18)$$

where $w_i \in R^{d_{model}}$ is a learnable weight vector. We use LeakRelu instead of Relu activation to resolve the problem of gradients fading during training. The softmax function is performed on each A, which can ensure the learning soft, neighboring matrix. The following property is satisfied by A

$$\sum_{j=1}^{N} A_{ij} = 1, A_{ij} \geq 0. (19)$$

We use the changed loss function based on **[10]** to optimize the w_i learning weight vector as

$$L_{GL} = \frac{1}{N^2} \sum_{i,j=1}^{N} exp(A_{ij} + \eta \left\| v_i - v_j \right\|_2^2 + \gamma ||A||_F^2), (20)$$

where $||.||_F$ represents *Frobenius-Norm.* Intuitively, the first item means that nodes $v_i$and v_j are far apart in higher dimensions encouraging a smaller weight value A_{ij}, and the exponential operation can enlarge this effect. Likewise, nuts closer to each other can have stronger connection weights in higher dimensional space. This process can avoid aggregation of the noise node information by graph convolution. η is a tradeoff parameter that checks the importance of graph nodes. The losses are also averaged because the number of nodes on the various documents is dynamic. The second item controls soft adjacent matrix A sparsity. γ is a tradeoff parameter, with a larger γ value resulting in a more sparse soft adjacent matrix A of graph. As shown in Eq.18 to avoid trivial solutions,. i.e., w_i= 0 as discussed in **[10]**, we use L_{GL} as a regularized term in our final loss function

Graph Convolution

Graph convolutional network (GCN) is being used to collect global visual information and the layout of the graph nodes. The the *node-edge-node* triplets (v_i, α_{ij}, v_j) as used in **[19]** is a graphic convolution instead of the node v_i alone.

Firstly, given an input $V^0 = X^0 \in R^{Nxd_{model}}$as the initial layer input of the graph, initial relation embedding α^0_{ij} between the node $v_i$and $v_j$is formulated as follows

$$\alpha^0_{ij} = W^0_\alpha [x_{ij}, y_{ij}, \frac{w_i}{h_i}, \frac{h_j}{h_i}, \frac{w_j}{h_i}, \frac{T_j}{T_i}]^T , (21)$$

where $W^0_\alpha \in R^{d_{model} x\, 6}$is a learnable weight matrix. The horizontal and vertical distances between the nodes v_i and v_j are called $x_{ij}$and , respectively.The width and height of the v_i and v_j node is w_i, h_i , w_j , h_j. The aspect ratio of node v_i is $\frac{w_i}{h_i}$, and $\frac{h_j}{h_i}$, $\frac{w_j}{h_i}$has affine invariance and uses the height of node $v_i$for normalization.

In addition, the $\frac{T_j}{T_i}$ length ratio between the v_i and v_j nodes are different to **[19]**. The length of the sentence intuitively contains information of latent importance. As an example, the age value entity for medical invoices is usually only three digits, which plays an important role in improving performance of key information extraction. Furthermore, the model can produce rough font sizes of text segments, given the length of the sentence and image, making the relationship embedding more representable.
Then, using the node-edge-node triplets (v_i, α_{ij}, v_j) data in the $l-th$ convolution layer, we retrieve hidden features h^l_{ij} between the nodes v_i and v_j from the graph, and h^l_{ij} is computed by

$$h^l_{ij} = \sigma(W^l_{v_ih} v^l_i + W^l_{v_jh} v^l_j + \alpha^l_{ij} + b^l), (22)$$

where $W^l_{v_ih}$, $W^l_{v_jh} \in R^{d_{model} x\, d_{model}}$ are the learnable weight matrices in the *l-th* convolution layer, and $b^l \in R^{d_{model}}$ is a bias parameter. $\sigma(\cdot) = max(0, \cdot)$ is a non-linear activation function. Hidden features $h^l_{ij} \in R^{d_{model}}$ are the total of image elements and the relation embedding between the nodes$v_i$and v_j, and they are essential for collating more richer representations for downstream tasks.
Finally, node embedding v^{l+1}_i uses graph convolution to update node representation by aggregating information from hidden features h^l_{ij}.In the same way that a graph learning layer can obtain an optimal adaptive graph soft adjacent matrix A, graph convolution layers can obtain task-specific node embedding by applying the layer-wise propagation rule. For node v_i, we have

$$v^{(l+1)}_i = \sigma(A_i h^l_i W^l)\ (23)$$

where $W^l \in R^{d_{model} x\, d_{model}}$ is a layer-specific learnable weight matrix in the l-th convolution layer, and $v^{(l+1)}_i \in R^{d_{model}}$ donates the node embedding for node v_i in the l + 1-th convolution layer. After L layers, we can get a contextual information v^L_i containing global layout information and visual information for every node v_i. Then v^L_i is propagated to the decoder for tagging tasks.
The relation embedding α^{l+1}_{ij} in the l+1-th convolution layer for node vi is formulated as

$$\alpha^{l+1}_{ij} = \sigma(W^l_\alpha h^l_{ij}), (24)$$

where $W^l_\alpha \in R^{d_{model} x\, d_{model}}$is layer-specific trainable weight matrix in the l-th convolution layer.

3.6.4. Decoder

Figure 3.6. depicts a decoder that includes the BiLSTM **[40]** layer, CRF **[41]**, and Union layer layer for key information extraction. Union layer receives the input X $\in R^{N\,x\,T\,x\,d_{model}}$ having variable length T generated from *Encoder*, then packs padded input sequences and fill padding value at the end of sequence yielding packed sequence $X_{hat} \in R^{(N \cdot T)\,x\,d_{model}}$. When performing sequence tagging with CRF, packed sequence X_{hat} can be regarded as the union nonlocal document representation rather than local text segment representation. Furthermore, at each time stamp we configured the node integration of the output of the Graph Module with the X_{hat}package sequence. Node embedding, which contains the layout of documents as well as contextual features as auxiliary information, appears to improve extraction performance without ambiguity. BiLSTM can use information from the context past/left and future/right as final output. The output of BiLSTM is given by

$$Z = BiLSTM(X_{hat};\ 0,\ \theta_{lstm})\, W_z, \quad (25)$$

where the result of BiLSTM is $Z = [z_1, ..., z_{N \cdot T}]^{N \cdot T} \in R^{(N \cdot T)\,x\,d_{output}}$, which represents the scores of the emissions matrix, $d_{output}$is the number of different entity, $Z_{t,j}$ represents the score of the *j*-th entity of the *t*-th character $c_t$in packed sequence X_{hat}, *0* means the initial hidden state and is zero, and θ_{lstm} represents the BiLSTM parameters.$W_z \in R^{d_{model}\,x\,d_{output}}$ is the trainable weight matrix.

Given a packed sequence X_{hat} of predictions y, its scores can be defined as follows

$$S(X_{hat}, y) = \sum_{i=0}^{N \cdot T} T_{y_i, y_{i+1}} + \sum_{i=1}^{N \cdot T} Z_{i, y_i}, \quad (26)$$

where $T \in R^{(N \cdot T+2)\,x\,(N \cdot T + 2)}$ is the scores of transition matrix and $y = (y_1, ..., y_{N \cdot T})$. The 'SOS' and 'EOS' entities of a sentence are represented by $y_0$and $y_{N \cdot T + 1}$, which stand for start of sequence and end of sequence, respectively. Then the sequence CRF layer generates a family of conditional probability via a softmax for the sequence y given X_{hat} as follows

$$p(y|X_{hat}) = \frac{e^{s(X_{hat}, y)}}{\sum_{y_{hat} \in y(X_{hat})} e^{s(X_{hat}, y_{hat})}}, \quad (27)$$

where $y(X_{hat})$ is all possible entity sequences for X_{hat}.

We reduce the negative log-likelihood estimate of the right entity series for CRF training, which is provided by

$$\begin{cases} L_{crf} = -\, log(p(y|X_{hat})) = -\, s(X_{hat}, y) + Z \\ Z = log(\Sigma_{y_{hat} \in y(X_{hat})} e^{s(X_{hat}, y_{hat})} = logadd_{y_{hat} \in y(X_{hat})} s(X_{hat}, y_{hat}) \end{cases} ,(28)$$

The following loss function is used to jointly train the model parameters of whole networks:

$$L_{total} = L_{crf} + \lambda L_{GL} \text{ , (29)}$$

where L_{GL} and L_{crf} are defined in Eq.20 and Eq.28 individually, and λ is a tradeoff parameter.
The output sequence y* with the highest conditional probability is searched during CRF layer decoding

$$y^{*} = argmax_{y_{hat} \in y(X_{hat})} p(y_{hat}|X_{hat}). \text{ (30)}$$

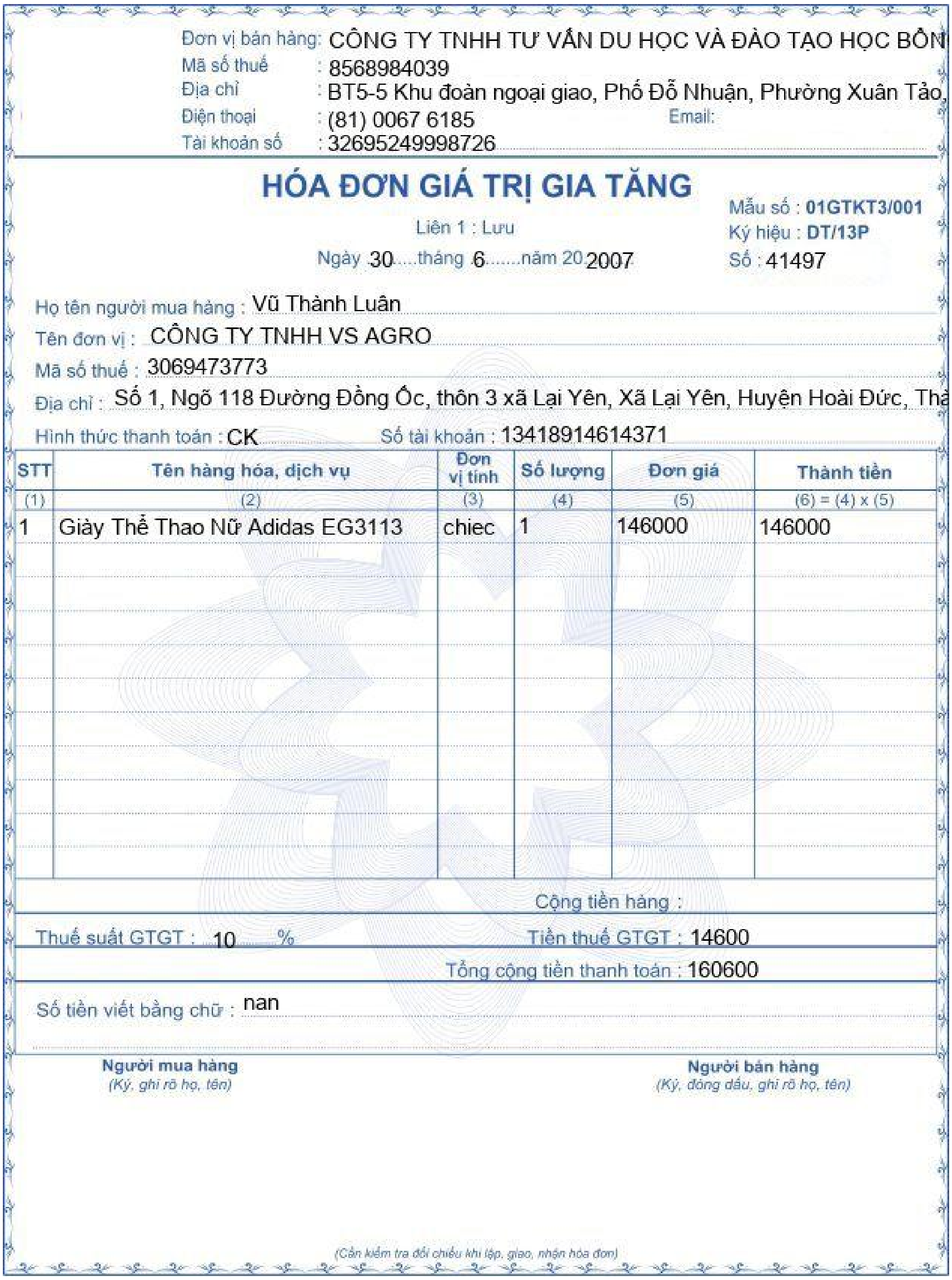
The training (Eq.28) and decoding (Eq.30) phases take time, but we can speed things up by using the dynamic programming algorithm.

CHAPTER 4. IMPLEMENTATION AND ANALYSIS

4.1. Dataset

The performance of the models is largely determined by the scale and quality of the data set. So we need a large, quality data set to train the model. Our model is trained by a data set of more than 2000 images created by us due to a lack of Vietnamese invoice data. Although we do generate the data on our own, they are perfectly suited for model training because we have relied on real data to generate them.

VAT invoice is a data set we created that contains more than 2000 pictures. Inside, 2000 pictures contain 1600 invoices for training and 400 invoices for testing. We took four different invoice templates to create our dataset. It has six key text fields including *sales company name, buyer's full name, tax, buyer's address, ATM, and total money*. This data set consists mainly of Vietnamese numerals and characters. Examples of VAT invoices are shown in the figure down here.



Đơn vị bán hàng:	CÔNG TY TNHH TƯ VẤN DU HỌC VÀ ĐÀO TẠO HỌC BỔN
Mã số thuế	: 8568984039
Địa chỉ	: BT5-5 Khu đoàn ngoại giao, Phố Đỗ Nhuận, Phường Xuân Tảo,
Điện thoại	: (81) 0067 6185 Email:
Tài khoản số	: 32695249998726

HÓA ĐƠN GIÁ TRỊ GIA TĂNG

Liên 1 : Lưu

Ngày 30.....tháng 6.......năm 20 2007

Mẫu số : **01GTKT3/001**
Ký hiệu : **DT/13P**
Số : 41497

Họ tên người mua hàng : Vũ Thành Luân
Tên đơn vị : CÔNG TY TNHH VS AGRO
Mã số thuế : 3069473773
Địa chỉ : Số 1, Ngõ 118 Đường Đồng Ốc, thôn 3 xã Lại Yên, Xã Lại Yên, Huyện Hoài Đức, Thà
Hình thức thanh toán : CK Số tài khoản : 13418914614371

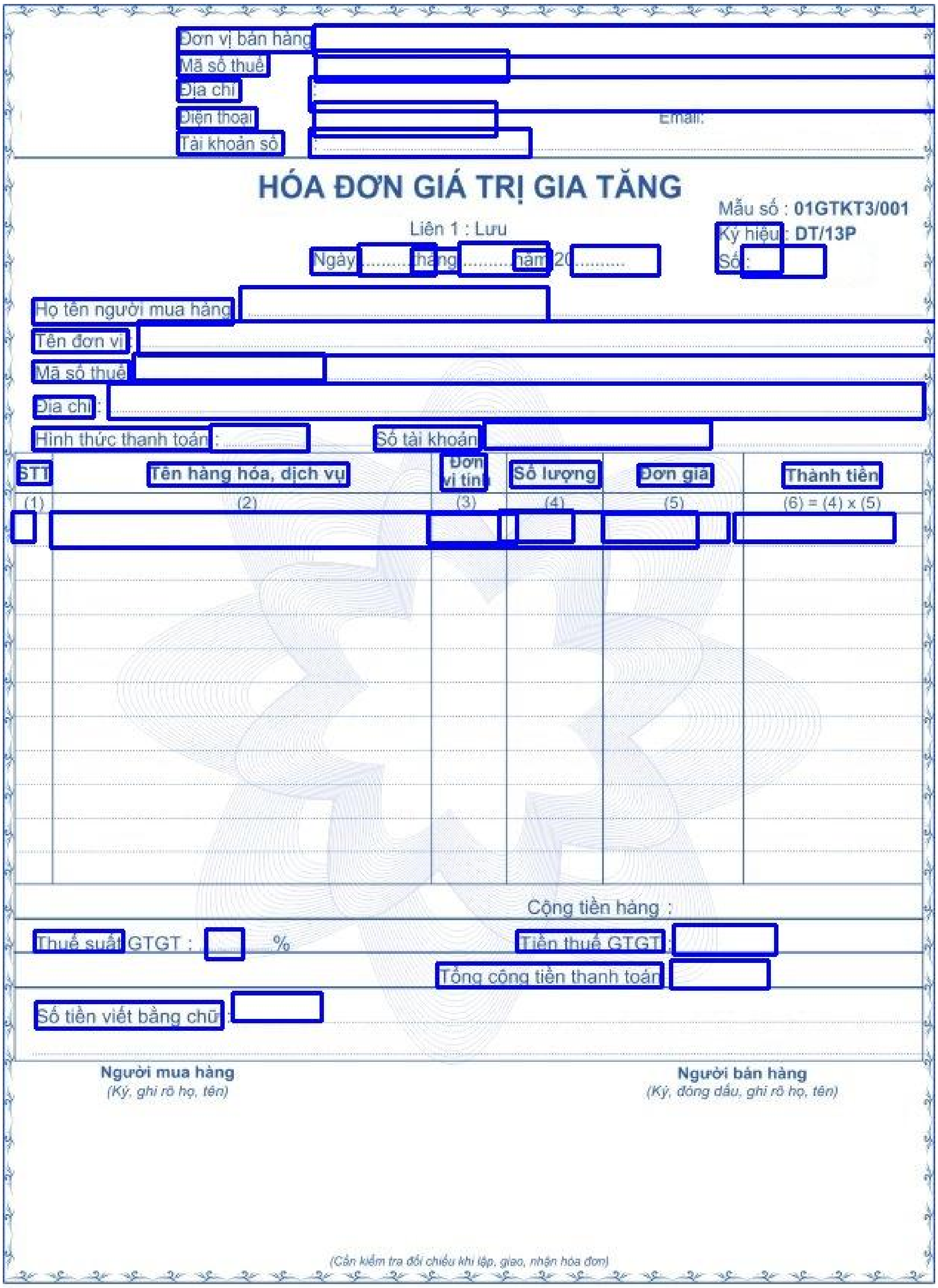
STT	Tên hàng hóa, dịch vụ	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
(1)	(2)	(3)	(4)	(5)	(6) = (4) x (5)
1	Giày Thể Thao Nữ Adidas EG3113	chiec	1	146000	146000

Cộng tiền hàng :
Thuế suất GTGT : 10 % Tiền thuế GTGT : 14600
Tổng cộng tiền thanh toán : 160600
Số tiền viết bằng chữ : nan

Người mua hàng	Người bán hàng
(Ký, ghi rõ họ, tên)	*(Ký, đóng dấu, ghi rõ họ, tên)*

(Cần kiểm tra đối chiếu khi lập, giao, nhận hóa đơn)

Figure 4.1.a. The VAT invoice has been processed



Đơn vị bán hàng
Mã số thuế
Địa chỉ :
Điện thoại   Email:
Tài khoản số :

HÓA ĐƠN GIÁ TRỊ GIA TĂNG

Liên 1 : Lưu

Ngày .... tháng .... năm 20..........

Mẫu số : **01GTKT3/001**
Ký hiệu : **DT/13P**
Số :

Họ tên người mua hàng
Tên đơn vị
Mã số thuế
Địa chỉ :
Hình thức thanh toán : ....   Số tài khoản

STT	Tên hàng hóa, dịch vụ	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
(1)	(2)	(3)	(4)	(5)	(6) = (4) x (5)

Cộng tiền hàng :
Thuế suất GTGT :   %   Tiền thuế GTGT :
Tổng cộng tiền thanh toán :
Số tiền viết bằng chữ :

Người mua hàng
(Ký, ghi rõ họ, tên)

Người bán hàng
(Ký, đóng dấu, ghi rõ họ, tên)

(Cần kiểm tra đối chiếu khi lập, giao, nhận hóa đơn)

Figure 4.1.b: VAT invoice we used Ybat tool to get the text box

Implementation Details

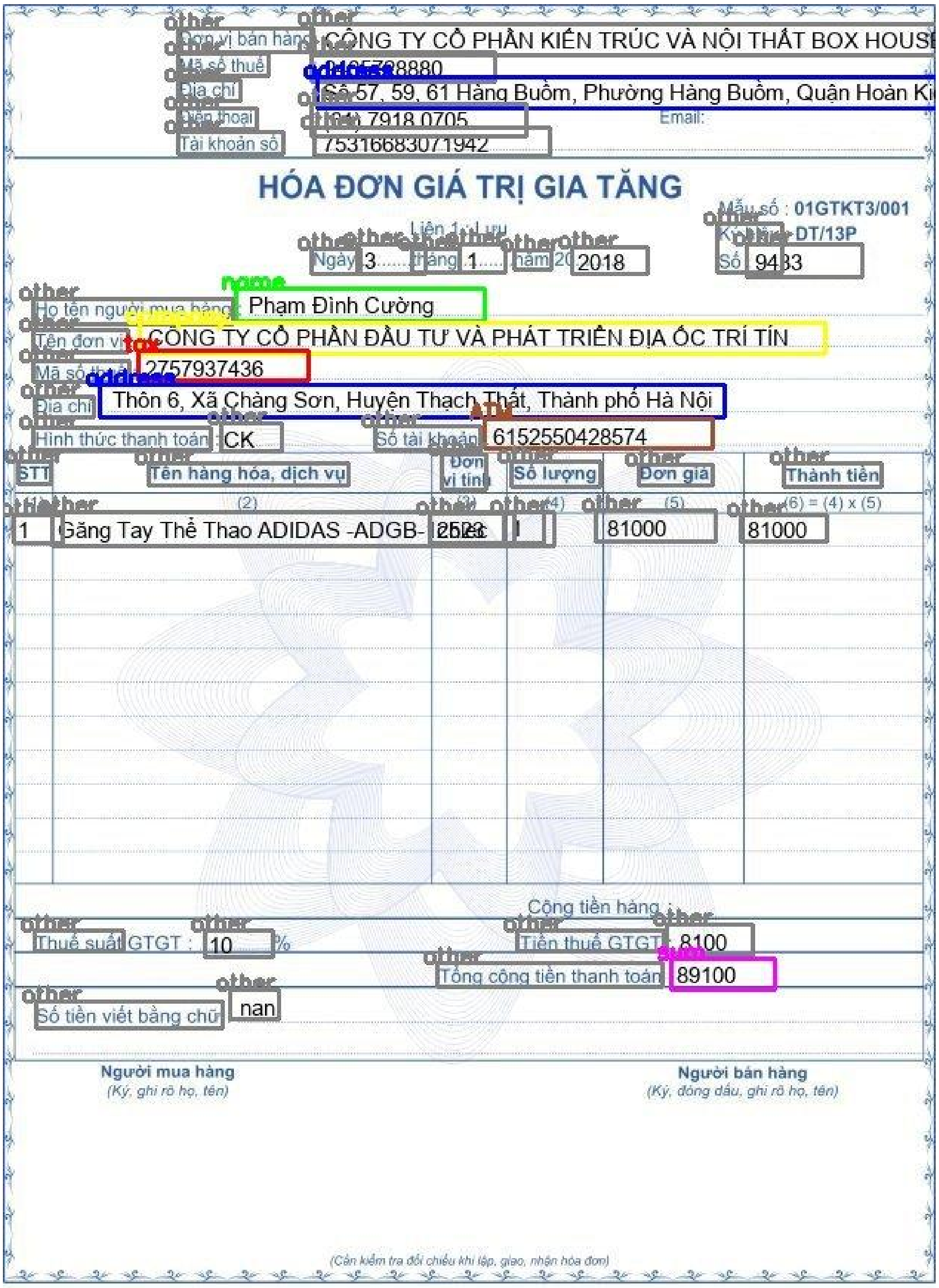
To create the desired data, we must have images, text and boxes. Here are the specifics of our procedure:

- **Step 1.** We collect sample value-added invoices. It is blank, no content..
- **Step 2.** For the image we already have, we use Ybat tool to get the box of text. Here we get the
- **Step 3.** coordinates of the box (x, y, width, height). Example of VAT invoice we used Ybat tool to get the text box, it is shown in figure 4.1.b.
- **Step 4.** We use our created tool to get relevant data in the content of the invoice. And put them and each of the different fields in the excel file.

With the existing box and content. We create our VAT invoice data.

An example of a VAT invoice that has been processed by us is shown in figure 4.1.b., and figure 4.1.b.

We use **metrics for evaluation** suggested by Guo, He. et al. (2019) [4] to evaluate the effectiveness of the model. Metrics are mean entity precision (mEP), mean entity recall (mER), and mean entity F-measure (mEF).


other
address
name
company
tax
acc
sum

Đơn vị bán hàng: CÔNG TY CỔ PHẦN KIẾN TRÚC VÀ NỘI THẤT BOX HOUSE

Mã số thuế: 0105728880

Địa chỉ: Số 57, 59, 61 Hàng Buồm, Phường Hàng Buồm, Quận Hoàn Ki

Điện thoại: (04) 7918 0705 Email:

Tài khoản số: 75316683071942

HÓA ĐƠN GIÁ TRỊ GIA TĂNG

Mẫu số : 01GTKT3/001

Ký hiệu: DT/13P

Liên 1 : Lưu

Ngày 3 tháng 1 năm 20 2018

Số 9483

Họ tên người mua hàng: Phạm Đình Cường

Tên đơn vị: CÔNG TY CỔ PHẦN ĐẦU TƯ VÀ PHÁT TRIỂN ĐỊA ỐC TRÍ TÍN

Mã số thuế: 2757937436

Địa chỉ: Thôn 6, Xã Chàng Sơn, Huyện Thạch Thất, Thành phố Hà Nội

Hình thức thanh toán: CK Số tài khoản: 6152550428574

STT	Tên hàng hóa, dịch vụ	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
(1)	(2)	(3)	(4)	(5)	(6) = (4) x (5)
1	Găng Tay Thể Thao ADIDAS -ADGB-12523	Chiếc	1	81000	81000

Cộng tiền hàng :

Thuế suất GTGT : 10 % Tiền thuế GTGT: 8100

Tổng cộng tiền thanh toán: 89100

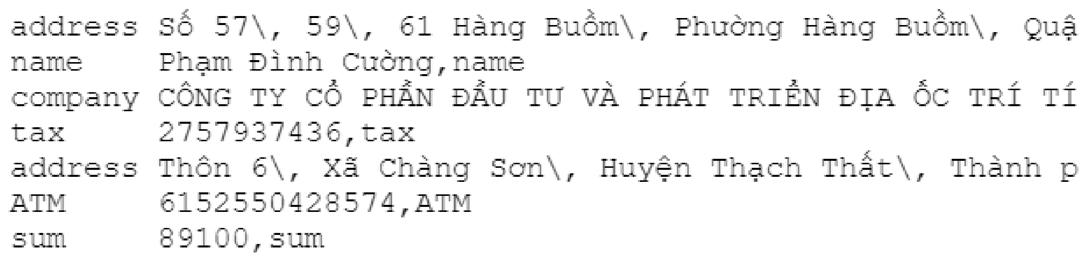
Số tiền viết bằng chữ: nan

Người mua hàng
(Ký, ghi rõ họ, tên)

Người bán hàng
(Ký, đóng dấu, ghi rõ họ, tên)

(Cần kiểm tra đối chiếu khi lập, giao, nhận hóa đơn)

Figure 4.1.c: The VAT invoice have been defined bbox and labeled



```
address Số 57\, 59\, 61 Hàng Buồm\, Phường Hàng Buồm\, Quậ
name    Phạm Đình Cường,name
company CÔNG TY CỔ PHẦN ĐẦU TƯ VÀ PHÁT TRIỂN ĐỊA ỐC TRÍ TÍ
tax     2757937436,tax
address Thôn 6\, Xã Chàng Sơn\, Huyện Thạch Thất\, Thành p
ATM     6152550428574,ATM
sum     89100,sum
```

Figure 4.1.d. Output of Figure 4.1.c.

4.2. Results

In this project, we used google colaboratory with GPU K80s to optimize the loss function of Graph learning, the loss function of CRF. With 1600 images of resize 960*720 and 15 epochs, we trained the model over about 12 hours and gave results in Table 4.2.

Entities	**mEP**	**mER**	**mEF**
ATM	0.917506	0.899287	0.908396
tax	0.915836	0.912856	0.914346
address	0.957516	0.912541	0.935028
name	0.934924	0.915429	0.925176
sum	0.906892	0.927898	0.917395
company	0.901946	0.926472	0.914209
Overall	**0.922436**	**0.915747**	**0.919091**

Table 4.2. Result in VAT data by mEP(mean entity prediction), mER(mean entity recall), mEF((mean entity F1).

Our model achieved better results than we expected on three indicators. MEP is around 92.2%, MER is around 91.5%, and MEF is around 91.9%. Further analysis reveals that when combined with the context of words, images, and coordinates (boxes), our model learns the correlation of irregular data and graph data. The graph model best illustrates the primary key "address" with very extensive sentence lengths and coordinates. Thus, it receives a very high MEF score of 93.5 percent due to the very distinctive element of the primary key "address". Two keys with a short length of change, "ATM" and "tax," represented as one-digit numbers, are predicted to be relatively accurate. However, the context of these two keys does not make much sense. So, the prediction of these two keys is 1-2 percent worse than the other primary key.

CHAPTER 5. CONCLUSIONS

We have successfully implemented a model for using contextual information, visual information, and textual coordination. In particular, The model uses the self-attention feature of the Transformer model to extract contextual features of the sentence itself, extract image features using the Convolutional neural network, and use the boxes-shaped coordinates features, distance, length. Then, the model has combined that factor to give a realistic view of Key Information Extraction(KIE) for text, especially Vietnamese. Therefore, the labeling for the documented information was more specific about the training method.

REFERENCES

[1] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016). *Neural Architectures for Named Entity Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1603.01360.

[2] Ma, X. and Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv:1603.01354 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1603.01354.

[3] Chiu, J.P.C. and Nichols, E. (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4, pp.357–370.

[4] guo, H., Qin, X., Liu, J., Han, J., Liu, J. and Ding, E. (2019). EATEN: Entity-aware Attention for Single Shot Visual Text Extraction. *arXiv:1909.09380 [cs]*. [online] Available at: https://arxiv.org/abs/1909.09380.

[5] Katti, A.R., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Höhne, J. and Faddoul, J.B. (2018). Chargrid: Towards Understanding 2D Documents. *arXiv:1809.08799 [cs]*. [online] Available at: https://arxiv.org/abs/1809.08799.

[6] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F. and Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, [online] pp.1192–1200. Available at: https://arxiv.org/abs/1912.13318 [Accessed 24 Apr. 2021].

[7] Swampillai, K. and Stevenson, M. (2011). *Extracting Relations Within and Across Sentences*. [online] ACLWeb. Available at: https://www.aclweb.org/anthology/R11-1004/ [Accessed 24 Apr. 2021].

[8] Rusinol, M., Benkhelfallah, T. and dAndecy, V.P. (2013). Field Extraction from Administrative Documents by Incremental Structural Templates. *2013 12th International Conference on Document Analysis and Recognition*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/6628784 [Accessed 5 Jan. 2020].

[9] Lebourgeois, F., Bublinski, Z. and Emptoz, H. (1992). *A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/201771/similar#similar [Accessed 24 Apr. 2021].

[10] Kipf, T.N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1609.02907.

[11] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. (2018). Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1710.10903.

[12] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J. and Bronstein, M.M. (2016). Geometric deep learning on graphs and manifolds using mixture model CNNs. *arXiv:1611.08402 [cs]*. [online] Available at: https://arxiv.org/abs/1611.08402.

[13] Song, L., Zhang, Y., Wang, Z. and Gildea, D. (2018). N-ary Relation Extraction using Graph State LSTM. *arXiv:1808.09101 [cs]*. [online] Available at: https://arxiv.org/abs/1808.09101.

[14] Peng, N., Poon, H., Quirk, C., Toutanova, K. and Yih, W. (2017). Cross-Sentence N-ary Relation Extraction with Graph LSTMs. *arXiv:1708.03743 [cs]*. [online] Available at: https://arxiv.org/abs/1708.03743.

[15] Wang, S., Zhang, Y., Che, W. and Liu, T. (n.d.). *Joint Extraction of Entities and Relations Based on a Novel Graph Scheme*. [online] . Available at: http://ir.hit.edu.cn/~car/papers/ijcai18slwang.pdf [Accessed 24 Apr. 2021].

[16] Marcheggiani, D. and Titov, I. (2017). Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. *arXiv:1703.04826 [cs]*. [online] Available at: https://arxiv.org/abs/1703.04826.

[17] Gui, T., Zou, Y., Zhang, Q., Peng, M., Fu, J., Wei, Z. and Huang, X. (2019). A Lexicon-Based Graph Neural Network for Chinese NER. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[18] Qian, Y., Santus, E., Jin, Z., Guo, J. and Barzilay, R. (2019). GraphIE: A Graph-Based Framework for Information Extraction. *arXiv:1810.13083 [cs]*. [online] Available at: https://arxiv.org/abs/1810.13083.

[19] Liu, X., Gao, F., Zhang, Q. and Zhao, H. (2019). *Graph Convolution for Multimodal Information Extraction from Visually Rich Documents*. [online] arXiv.org. Available at: https://arxiv.org/abs/1903.11279.

[20] LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

[21] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S. (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, [online] pp.1–21. Available at: https://ieeexplore.ieee.org/abstract/document/9046288 [Accessed 2 Oct. 2020].

[22] Schaeffer, S.E. (2007). Graph clustering. *Computer Science Review*, [online] 1(1),

pp.27–64. Available at:
http://www.leonidzhukov.net/hse/2015/networks/papers/GraphClustering_Schaeffer07.pdf.

[23] Kipf, T.N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1609.02907.

[24] Jiang, B., Zhang, Z., Lin, D. and Tang, J. (2018). Graph Learning-Convolutional Networks. *arXiv:1811.09971 [cs]*. [online] Available at: https://arxiv.org/abs/1811.09971.

[25] Zhang, Y., Chen, G., Yu, D., Yao, K., Khudanpur, S. and Glass, J. (2016). Highway Long Short-Term Memory RNNs for Distant Speech Recognition. *arXiv:1510.08983 [cs, eess]*. [online] Available at: https://arxiv.org/abs/1510.08983.

[26] Graves, A., Mohamed, A. and Hinton, G. (2013). *Speech Recognition with Deep Recurrent Neural Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1303.5778.

[27] Graves, A. (2012). *Sequence Transduction with Recurrent Neural Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1211.3711.

[28] LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521(7553), pp.436–444.

[29] Greenstein, E. and Penner, D. (n.d.). *Japanese-to-English Machine Translation Using Recurrent Neural Networks*. [online] . Available at: https://cs224d.stanford.edu/reports/GreensteinEric.pdf [Accessed 24 Apr. 2021].

[30] Cho, K., Merrienboer, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. [online] arXiv.org. Available at: https://arxiv.org/abs/1406.1078.

[31] Cho, K., Merrienboer, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. [online] arXiv.org. Available at: https://arxiv.org/abs/1406.1078.

[32] Sutskever, I. and Hinton, G. (2010). Temporal-Kernel Recurrent Neural Networks. *Neural Networks*, 23(2), pp.239–243.

[33] Karpathy, A. and Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. *arXiv:1412.2306 [cs]*. [online] Available at:

https://arxiv.org/abs/1412.2306#:~:text=Deep%20Visual%2DSemantic%20Alignments%20for%20Generating%20Image%20Descriptions.

[34] Karpathy, A. and Fei-Fei, L. (2017). Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 39(4), pp.664–676. Available at: https://cs.stanford.edu/people/karpathy/cvpr2015.pdf [Accessed 13 Dec. 2019].

[35] Chen, G. (2018). A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation. *arXiv:1610.02583 [cs]*. [online] Available at: https://arxiv.org/abs/1610.02583.

[36] Sutton, C. and McCallum, A. (2010). An Introduction to Conditional Random Fields. *arXiv:1011.4088 [stat]*. [online] Available at: https://arxiv.org/abs/1011.4088.

[37] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, Aidan N, Kaiser, L. and Polosukhin, I. (2017). *Attention Is All You Need*. [online] arXiv.org. Available at: https://arxiv.org/abs/1706.03762.

[38] Bahdanau, D., Cho, K. and Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.0473.

[39] He, K., Zhang, X., Ren, S. and Sun, J. (2015). *Deep Residual Learning for Image Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1512.03385.

[40] Graves, A. and Schmidhuber, J. (2005). *Framewise phoneme classification with bidirectional LSTM networks*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/1556215 [Accessed 27 Apr. 2021].

[41] Lafferty, J., McCallum, A. and Pereira, F. (2001). *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. [online] undefined. Available at: https://www.semanticscholar.org/paper/Conditional-Random-Fields%3A-Probabilistic-Models-for-Lafferty-McCallum/f4ba954b0412773d047dc41231c733de0c1f4926 [Accessed 27 Apr. 2021].

[42] Shanahan, J.G. and Roma, N. (2003). Improving SVM Text Classification Performance through Threshold Adjustment. *Machine Learning: ECML 2003*, pp.361–372.

[43] Kipf, T.N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1609.02907.