A Deep Learning-Based Approach for Named Entity Recognition on Commercial Receipts

by

Spencer Runshuo Han

Dissertation Submitted in Fulfillment of the
Requirements for the Degree of
Master of Professional Studies in Data Science

School of Computer Science
Faculty of Science

UNIVERSITY OF AUCKLAND
S2 2020

Table of Content

List of Figures

Abstract

Transaction management and expense analytics service providers increasingly seek to establish an automated system to harness information from commercial receipts. With the recent enhancement in the area of optical character recognition, the transformation from digital invoice documents to invoice text data is made possible and reliable. It is now becoming a priority to extract and identify essential entity information from the transformed invoice text data.

In this affiliated study with Fraedom Limited, we proposed a deep learning-based named entity recognition method to identify named entities from Amazon Textract processed commercial receipts text data.

We begin with preprocessing the transformed invoice text data to create labelled training, validation and testing dataset. And then we developed three deep learning architectures based on the concept of Bi-directional Long Short-term Memory Networks, Convolutional Neural Networks, Conditional Random Fields and Softmax. We then evaluated the three architectures according to their training performance against our validation dataset and testing dataset. Lastly, we provide a working example of solving a named entity recognition task using our best model as well as recommendations for future work in this area.

Our research has shown that with the combination of deep contextualised word representations, bi-directional long-short term memory networks and softmax classifiers, our proposed deep learning method is capable of solving the named entity recognition challenge on unstructured invoice text under a supervised machine learning scheme. However, we have also discovered possible areas for future improvement.

Keywords: Natural Language Processing; Named Entity Recognition; Optical Character Recognition; Deep Learning; Recurrent Neural Network; Convolutional Neural Network; Conditional Random Fields; Softmax; Deep contextualised word representations; ELMo

Acknowledgements

I would like to take this opportunity to thank my supervisor Sebastian Link for his guidance, encouragement and support throughout my dissertation. I am also grateful to Fraedom NZ Ltd. for providing this research opportunity. A great appreciation goes to David Duan (Principle Data Scientist) and Casper Hart (Data Scientist) at Fraedom, for providing project supervision as well as insights and feedback on my research methodologies from time to time. Without their supports, I may not be able to set the correct direction for this project and overcome technical challenges that were faced during implementing natural language processing approaches.

1

Introduction

With the prefoliation of image capturing devices and document scanning application, a tremendous amount of text data from commercial receipts and invoices are captured every day. Having an automated method that is capable of harnessing business information from such text data is a fundamental cornerstone for transaction management and expense analytics. As text data are natively stored in a variety of digital formats, much of the process to extract the critical text information from them is carried out manually or at least partially manually. Nevertheless, with the recent advancement in the area of deep learning for computer vision, state of arts of neural networks has drastically improved the accuracy of Optical Character Recognition (OCR). The success of a number of enterprise OCR applications such as Amazon Textract, Google Vision and Microsoft Azure Vision [1] have proven the reliability of automated text recognition from digital images. It has now become a natural language processing (NLP) task to identify and extract structured information from OCR processed text data.

One of the popular techniques of extracting information from text data is known as Named Entity Recognition (NER). As an area within the broader realm of NLP, NER aims to segment specific linguistic phrases (entities) appeared in text and categorised these linguistic phrases (entities) with various predefined classes such as organisation, person, date and price. For a long time, NER tasks are leveraged by the exploitation of data corpora, well constructed lexical patterns and 'shallow' machine learning models [2]. Although this approach has shown excellence in coping with large-scale NER challenges as in OntoNote v4, MUC-7 and CoNLL-2003 [3] [4] [5], it lacks the capability to discover hidden and unseen features within text data. As a result, the performance and robustness of the traditional approach are far from human-level performance when it comes down to NER tasks [2].

Unlike early NER approaches that rely heavily on domain knowledge and feature engineering, deep learning-based NER methods are capable of learning high-level abstraction of semantic features without prior knowledge and designing NER features. A number of state-of-art deep learning NER frameworks have been invented in recent years. They are usually built upon a collection of convolutional neural network (CNN), recurrent neural network (RNN), conventional probabilistic model (conditional random fields (CRF)) and advanced word embeddings method [6] [7] [8] [9]. Nevertheless, these deep learning-based NER approaches have traditionally focused on processing structured text documents that inherently incorporate syntactic rules. In contrast, commercial receipts exist in an unstructured form factor and do not necessarily possess prevailing grammar patterns [10]. Although some studies have focused on evaluating the performance of deep learning models in NER tasks against unstructured text information such as medical and biomedical datasets [11] [12], their abilities to identify key entities within OCR-ed business receipt data is not well explored.

This paper presents a deep learning-based approach to identify named entities (NE) from OCR'ed commercial receipts data. The main research goal is to demonstrate the feasibility of applying state-of-the-art deep learning models on unstructured invoice text data in a NER task. In the following sections, this paper will firstly provide detailed background and objectives of our study; then related work and literature will also be presented to support our approach; thirdly this paper will explain our principal methodology, including the data preprocessing pipeline and model implementation; Lastly, we will illustrate model evaluation result alongside with further discussions which we hope to offer insights for similar NER tasks in the future.

2

Background and Objectives

The company Fraedom Ltd. is a Visa-owned global information service company that specialised in 'Business to Business' payment solution, transaction management and expense reporting and analytics. Currently, Fraedom holds large quantities of business transaction and expense documents such as goods and service tax (GST) receipts and invoices. These documents are converted from original image format to text format with the Amazon Textract OCR engine. Hence, it is now an opportunity to apply NLP, more precisely, NER practice to extract specific entity information appeared in these documents. However, there are specific research questions are addressed additional to the overall goal:

1. First, most state-of-art NER deep learning architectures or frameworks are developed and examined on structured natural language data. They are trained and tested on data that possess human language syntactic feature and obeys specific grammatical rules. However, business transactions and invoices are mostly unstructured and do not necessarily follow these rules. The most fundamental question relates to the overall aim of this project is that is it still applicable to use such NER models extract entities from unstructured text data? Or in other words, can a state-of-art deep learning model successfully extract entities from unstructured text data?

2. Secondly, OCR'ed output data, by nature, contain a high degree of noises. Noises such as misreadings can be introduced by distortions found in poorly captured invoice data and the weakness of the OCR tool, in this case, Amazon Textract, itself. As possible improvements on document image qualities and Amazon Texract are out of the scope of this project, some degree of noises in the input text data are

inevitable. Thus, the question of how well the state-of-art NER approaches in coping with data noises will be explored. For example, what is the prediction score when a model is trained and tested on data has higher quality? In contrast, what is the prediction score when a model is trained and tested on data has a relatively lower quality?

3. Thirdly, there are several state-of-art deep learning-based NER architectures have been invented by researchers in the recent time. They share some similarities but also possess unique features. This project will choose three NER architectures as candidates to explore how do they compare to each other in terms of their cost, performance and robustness. For example, what is their training time? What is memory consumption during training? What is their predictability? How the results differ when the amount of training, validation and testing data changes?

4. Lastly, this research project seeks to provide a repeatable workflow and future recommendation in the area of NER for business transaction data. Therefore, the areas of improvement and consideration will be provided. These discussions hopefully will guide others who want to implement a similar NER system.

3

Literature and Related work

3.1 Optical Character Recognition (OCR)

Optical Character Recognition is a process of identifying printed or handwritten alphanumeric or other text characters inside digital images of physical documents such as scanned document [13]. It is widely used as a technique to convert text data inside digital images to machine-readable data format. Dated back to 1950s, OCR process was initially carried out by using mechanical and optical means of rotating disks and photomultiplier, flying spot scanner with a cathode ray tube lens, followed by photocells and arrays of them [14]. By then, OCR operation is unreliable and time-consuming. Nevertheless, throughout years of the improvement not only in pattern recognition algorithms but also in image processing hardware, the robustness and reliability of OCR have been strengthened. Today, many large data and information technology providers such as Amazon, Google and Microsoft provide ready-to-use OCR services with a Service as a product (Saas)' scheme [1] so that OCR tasks can be executed by users with the only minimal development effort. Besides, there are open source OCR products such as the 'Tesseract OCR' [15] also enriches the OCR markets. Since the data used in this study is output from Amazon Textract, the background of Amazon Textract and its specification will be discussed with more details in the following section.

3.2 Amazon Textract

Amazon Textract is Amazon's bespoke cloud-based OCR service that can detect text in a variety of documents. It is developed and maintained by Amazon's machine learning and computer vision experts [16], thus there is no additional development of OCR engine needed for users who want to transform digital copies of documents into text data. When

performing document text detection operation, image(s) of document(s) are processed by Amazon Textract API, and then a JSON data is returned containing the following information: 1. The lines and words of detected text; 2.The relationships between the lines and words of detected text; 3.The page that the detected text appears on; 4.The location of the lines and words of text on the document page; 5. The confidence score of an OCR reading [16]. A sample digital tax invoice document and a part of the corresponding Amazon Textract output is demonstrated below (Figure 1 and 2).


THE SHOT CAFE
TAX INVOICE
GST:122-530-485
SALES:1004 DATE:22/10/2020 9:48 NO.:1
Invoice No:202010220046
==========
Cappuccino +Large+Soya $5.50
1 Items Total:$5.50

GRAND TOTAL $5.50
INCLUDE TAX $0.72
PAY BY EFTPOS:$5.5

Thanks and you are welcome again.

Figure 1: Sample invoice document as in image format

```
{
  "BlockType": "LINE",
  "Confidence": 99.98200225830078,
  "Text": "THE SHOT CAFE",
  "Geometry": {
    "BoundingBox": {
      "Width": 0.45483526587486267,
      "Height": 0.046412542462348944,
      "Left": 0.058733854442834854,
      "Top": 0.19915041327476501
    },
    "Polygon": [
      {
        "X": 0.058733854442834854,
        "Y": 0.19915041327476501
      },
      {
        "X": 0.5135691165924072,
        "Y": 0.19915041327476501
      },
      {
        "X": 0.5135691165924072,
        "Y": 0.24556295573711395
      },
      {
        "X": 0.058733854442834854,
        "Y": 0.24556295573711395
      }
    ]
  },
  "Id": "556f9b42-948a-45b9-93c3-1ff634053c19",
  "Relationships": [
    {
      "Type": "CHILD",
      "Ids": [
        "aa30c465-10dd-4e9c-80e6-a32ebb672920",
        "b10954a4-85e9-436c-8864-656148bced22",
        "5cb1718a-f8cc-4abb-ae4a-334e988d9622"
      ]
    }
  ]
},
```


Figure 2: Sample JSON output after processes with Amazon Textract OCR engine.

As illustrated in Figure 2 and 3, with Amazon Textract, the original digital documents are transformed into text files. It enables users to perform NLP tasks with these text information.

3.3 Natural Language Processing

Natural language, in general, is a linguistic system constructed specifically for the purpose to convey or deliver semantics. The novel practice of a natural language processing (NLP) task is to facilitate the use of machines to achieve the study of phonetics, phonology morphology, syntax, semantics and pragmatics of natural languages. Corresponding to the six NLP features, typical applications or tasks of NLP include speech recognition, spoken language understanding, text mining, part-of-speech analysis, machine translation and human-machine interaction.

In the early stage, the dominant strategy to accomplish NLP applications or tasks is to incorporate domain knowledge and reasoning mechanisms to design a collection of hand-crafted rules, for example, the pattern of date (YYYY/mm/dd and dd/mm/YYYY). With the hand-crafted rules, the machine emulates the understanding of a natural language by applying those rules to the data that the machine is confronted with. As rules are declarative, a rule-based NLP strategy is transparent and expressive. Such a strategy is foremost in the possibility of directly transferring domain knowledge into rules without the selection of training data, hyperparameters or weights [17]. Nevertheless, crafting rules requires extensive expert knowledge and manual input. Hence rule-based strategies are least applicable to people outside the demography of domain experts. Besides, because rules are declarative and manually crafted, it is often failing to cope with minor variations in the data, and also capturing all variety of vocabulary patterns are inevitably overwhelming.

Following the early NLP strategy, empirical and statistical NLP strategies started prevailing from the 1990s as a result of the exploitation of an immense amount of text data and corpus [18]. With statistical models, possessing domain-specific knowledge is no longer required for NLP specialists when solving NLP tasks. Ample investment in the effort of attempting to conclude all possible variations in linguistic patterns is also no longer mandatory. Hence, the focus of crafting rules shifts to tuning parameters of statistical models. In general, empirical NLP strategies perform much better than earlier domain knowledge-based NLP strategy, particularly in the area of capture postulate uncertainties. The key NLP algorithms used in this period include Linear Conditional Random Fields (CRFs), Bayesian networks, Support Vector Machines (SVM) and Hidden Markov Model (HMM).

Deep learning methods predominantly leverage recent NLP strategies. With empirical and statistical NLP strategy, 'shallow' models are trained on highly sparse datasets [19]. Additional manual effort in feature engineering often required to reduce the dimensionality of the training data. However, it is difficult to cover all regularities in languages by engineering features manually. In contrast, neural networks that are based on dense vector representations have produced superior results on various NLP tasks [19]. Originated from neural networks, deep learning methods are capable of extracting hidden features and

enables multi-level automatic feature representation learning. It vastly reduced the level of manual input on feature engineering and dependences on domain knowledge expertise. In this period, popular neural networks used in accomplishing NLP tasks are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Regardless of the category of NLP strategy in use, a pipeline of data preprocessing is necessary for the completion of an NLP task. A typical data preprocessing pipeline for an NLP task is illustrated below (Figure 3).

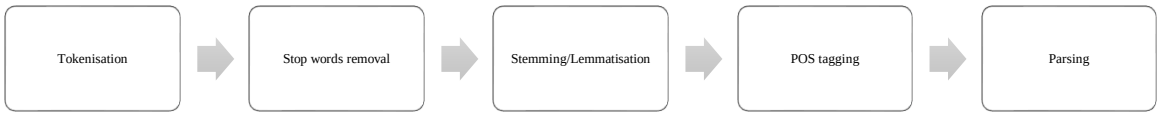


Figure 3: A typical data preprocessing pipeline in an NLP task.

Nonetheless, as natural language system and NLP tasks are incredibly sophisticated, variations in these preprocessing steps have been observed when NLP is applied in specific topics and areas of studies [20] [21] [22]. For example, techniques such as *token substitution* are used by some researchers [20] to reduce and normalise the complexity of input natural language data whereas the others [21] use *chucking* to group syntactically related tokens into smaller groups. Therefore, it is not necessary to follow all the steps outlined in figure 3 above. Data preprocessing should be situationally depending on the purpose of the NLP task.

3.4 Named Entity Recognition

Named entities (NEs) in a sentence or document are words or phrases which are referent of real world-object such as organisations, persons, products and amount. Named Entity Recognition (NER) seeks to locate and classify NEs mentioned in natural language data into predetermined categories. In a sample sentence 'Google announces the new Pixel Phone on 10 September 2019', the probable NEs are (highlighted in bold font):

organisation (Google), **product** (Pixel Phone) and **date** (10 September 2019).

In a real NER practice, there are different NE tagging systems which serve the same purpose for annotating NEs but with slight variations. One of the most popular NE tagging formats is the BIO system - a plausible synonym to inside-outside-beginning tagging [23]. Using the sample sentence above, a full BIO formatted NE tags annotated for the sample sentence can be seen as:

Token	BIO Tag
Google	B-ORG
announces	O
the	O
new	O
Pixel	B-PRODUCT
Phone	I-PRODUCT
on	O
10	B-DATE
September	I-DATE
2019	I-DATE
.	O

Figure 4: BIO NE tagged tokens in a sample sentence ('Google announces the new Pixel Phone on 10 September 2019.'.

In BIO tagging, the prefix 'B-'indicates the beginning of a chunk of an NE and the prefix 'I-' indicates the token is inside of a chunk of an NE [23]. An 'O' tag is used to indicate that a token does not belong to any NE [23]. It is evident that the transition from 'I-' tag to 'O' tag defines the tail end of an NE chunk [23].

As a subfield of NLP, NER methodologies are also defined in three broad categories: rule-based NER, statistical-based NER and deep learning-based NER. In a rule-based system, ranges of the corpus are annotated with NE tags. The annotated corpus forms NE dictionaries or domain-specific gazetteers. Unannotated natural language data are consequentially compared to the NE dictionaries and gazetteers based on a set of defined

lexical patterns (hence rules). Tokens that fulfils the matching criteria are assigned with the corresponding NE eventually. In contrast, statistical-based NER systems are governed by statistical probabilities. A sequence of natural language data can be represented by a series of token x_1,x_2,x_3…….x_n. Simultaneously, their corresponding NE label sequence (such as B-ORG, O, B-PRODUCT, I-PRODUCT) is expected to be y_1, y_2, y_3……..y_n. Thus in statistical-based approaches, NER tasks become a problem to find the NE label sequence that satisfies:

$$S = \underset{y_1 \dots y_n}{argmax}\, P(y_1 \dots y_n | x_1 \dots x_n)$$

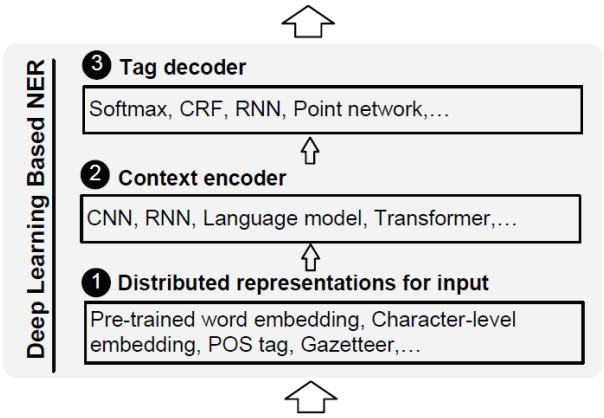
Applying the Bayes' theorem of probabilities, the above criteria can be rewritten as:

$$S = \underset{y_1 \dots y_n}{argmax}\, P(x_1 \dots x_n | y_1 \dots y_n)\, P(y_1 \dots y_n)$$

There are a number of popular statistical models used to solve NER challenges. One of them is the Conditional Random Fields (CRF) algorithm. A detailed background of CRF will be provided in the later section.

In addition to statistical modellings, deep learning-based NER methodologies have gained wide attention from academia and industrial practitioners. In summary, deep learning model excels in NER tasks in three aspects: first, unlike linear statistical models such as CRF, the input and output are non linearly mapped in deep learning models. This leads to deep learning models are capable of discovering complex and intricate hidden features from natural language data via complex non-learning activation functions; secondly, extensive feature engineering and domain-specific knowledge are necessities in the conventional approaches. Deep learning models, on the other hand, are effective in automatically extracting useful representation and intrinsic factors from raw natural language data; lastly, deep learning models inherit the features from neural networks. Hence deep learning models are flexible and elastic when adapting for different learning tasks via tuning the number of neurons and hidden layers [24]. Figure 5 below demonstrates a typical deep learning-based NER framework.

B-ORG	O	O	O	B-PRODUCT	I-PRODUCT	O	B-DATE	I-DATE	I-DATE	O
Google	announces	the	new	**Pixel**	**Phone**	on	**10**	**September**	**2019**	.



Google announces the new Pixel Phone on 10 September 2019.

*Figure 5: The taxonomy of a hybrid deep-learning NER for the sample sentence: 'Google announces the new Pixel Phone on 10 September 2019' *inspired by [24].*

3.5 Conditional Random Fields

The Conditional Random Fields (CRFs) is a popular class of classifier used in learning sequential data in which contextual information or state of neighbours affect the current prediction. In CRF, given that a sequence of natural language tokens X ($x_1 \dots x_n$) and a sequence of NE annotations Y ($y_1 \dots y_n$), a chain-structured case of CRF for a NER model can be deduced as in Figure 6, where 'i' is the position within the sample sequence, and an open circle indicates that the data is not generated by the model:

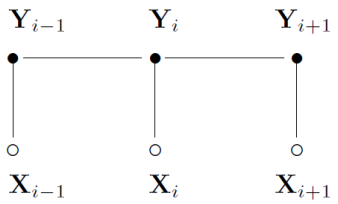


Figure 6: Graphic representation of a linear-chain CRF for natural language sequences [25].

Then the probability of calculating a label sequence Y given a token sequence X can be written as:

$$P_\theta(Y|X;\omega) = \frac{\exp\left(\sum_j \omega_j F_j(X,Y)\right)}{\sum_{Y'} \exp\left(\omega_j F_j(X,Y')\right)} \text{ where } F_j(X,Y) = \sum_{i=1}^{L} f_j(Y_{i-1}, X_i, X, i) \; {}^*\omega \; for \; weigths$$

The $F_j(X,Y)$ is the summation of values from a feature function for all tokens. Using a short subset of the earlier sample sentence 'new **Pixel Phone**' with NE tag [O, B-PRODUCT, I-PRODUCT], the numerator of the above equation is equal to:

$$\exp\left(\sum_j \omega_j \sum_i F_j([\text{'}new\ Pixel\ Phone\text{'}], \text{'O B} - \text{PRODUCT I} - \text{PRODUCT'})\right)$$

and the denominator is equal to the summation of all possible NE annotation combinations:

$$\exp\left(\sum_j \omega_j \sum_i F_j([\text{'}new\ Pixel\ Phone\text{'}], \text{'O B} - \text{PRODUCT I} - \text{PRODUCT'})\right) +$$

$$\exp\left(\sum_j \omega_j \sum_i F_j([\text{'}new\ Pixel\ Phone\text{'}], \text{'O O O'})\right) +$$

$$\exp\left(\sum_j \omega_j \sum_i F_j([\text{'}new\ Pixel\ Phone\text{'}], \text{'O I} - \text{PRODUCT I} - \text{PRODUCT'})\right) \ldots$$

Ultimately, for a well-trained model, the probability of ([O B-PRODUCT I-PRODUCT] | 'new **Pixel Phone**') will be highest among all possible combinations. To estimate the NE labels based on the input sequence token X, the regularised log-likelihood function for the probability distribution is:

$$L(\omega) = \sum_{i=1}^{n} \log p\left(y^i \middle| x^i; \omega\right) - \frac{\lambda_2}{2} |\omega|_2^2 - \lambda_1 |\omega|_1$$

Then the most likely NE tags set for a sequence of token X can be discovered by:

$$Y^* = \arg max_Y \, p(Y|X;\ \omega^*)$$

3.6 Softmax

As an alternative to the CRFs, the softmax function can also be used as a classifier to obtain the probability distribution over predicted NE classes. Unlike CRFs, softmax function takes as input a vector *z* of K real number and normalise it into a probability distribution consisting of K probabilities that proportion to the exponentials of the input numbers. After applying softmax, each component in the vector will be in the interval [0, 1]. From all possible classification outcomes, the one has the highest probability (closet to 1) will be the final predicted class. Below is the standard softmax function:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Even though the softmax is a pure probabilistic function and does not take into account contextual influences, studies have shown [24] that CRF does not guarantee a better result and there are scenarios the exponential function in softmax yields better results.

3.7 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a special type of feed-forward Artificial Neural Networks (ANNs) that were initially designed for image recognition tasks [26]. The primary purpose of CNNs is to have a neural network where the neurons in the early stage of the network would extract local features and the neurons in the later stage would combine these features to form higher-order features. When an input vector data (such as in X ×Y dimension) went through a CNN network, a small learning filter (kernel) is applied to capture the local features from a patch of adjacent data units (pixels). An output is produced by sliding the filter across the entire input data until all data units (pixels) are covered [27] (Figure 7).

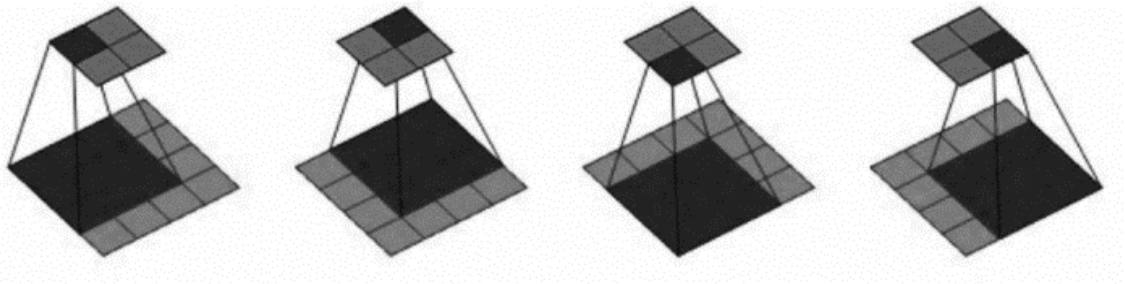


Figure 7: Convoling a 3 × 3 kernel (W) over a 4 × 4 input (X) to generate a 2 × 2 output (Y) [27].

By sliding filters (kernels) over the input data, CNNs reduces the total amount of information but with the purpose of retrieving only the relevant features by focusing on individual data regions. In a NER task, CNNs can be used to extract character level features of each word tokens. The input matrix is essentially the length of a sentence times all possible character candidates found in each character location within the sentence. Then by applying the same sliding filter strategy, it is now possible to retain the character level features of each word tokens in a sentence. The detailed usage of CNNs in NLP will be explained in the later methodology section.

3.8 Recurrent Neural Networks and Long Short-term Memory

In comparison to CNNs, recurrent neural networks (RNNs) serve as the foundation in processing sequential natural language data. In the domain of NLP, dependencies, or relationships can be found within the sequential data input. For instance, in the sentence 'Josh ___ to ___', filling the first blank with a verb such as 'fly' implies a very restricted set of values that are possible to appear in the second blank, wheres by changing the verb from 'flies' to 'learned', there will be an entirely new set of probable value for the second blank. Thus, having a deep learning architecture to capture the sequential dependencies or relationship is crucially important. An RNN network processes a set of sequential data by processing each element found in the sequential data one at a time. An RNN only contains a single hidden layer but has a memory buffer to store the output of this hidden layer from one input element and feeds it back into the hidden layer along with the next input element

from the sequence. By doing so, the context of information gained from processing the previous input element is passed onto the next processing stage [26]. When a recurrent flow is established for the entire set sequence data, a later processing stage within the RNN network contains the contextual information from all earlier input elements preceding it. Figure 8 illustrates how the information flows within an RNN network with crossreference to conceptual timesteps.

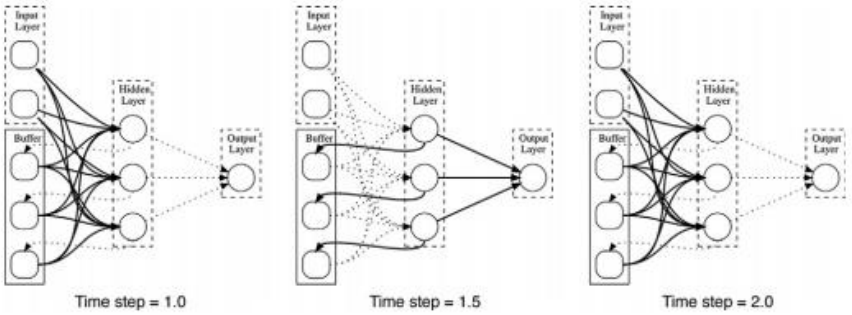


*Figure 8: The recurrent information flow in an RNN network as it processes a sequential input [26] * the bold arrow indicates active paths of information flow at each time step whereas the dashed arrows show connections that are not active at the time.*

When unrolling through time, an RNN network that processes a sequence of input $[x_1 \dots x_k]$ and generate a sequence of corresponding output $[h_1 \dots h_k]$ can be represented as in Figure 9:

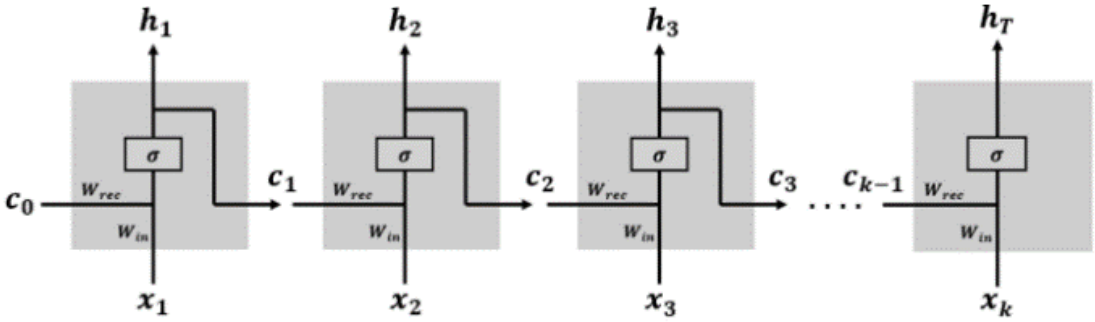


Figure 9: Unrolling an RNN network through time by creating a copy of the model for each time steps [26].

At time step t the network has an input vector x_t. Past information and learned knowledge is encoded in the network state vectors $[c_1 \dots c_{k-1}]$. At time step 1 the network has an input state vector c_{t-1}. The input vector x_t and the state vector c_{t-1} are concatenated to comprise the complete input vector at time step t, $[c_{t-1}, x_t]$. An RNN network also has two

weight matrices: recurrent weight (w_{rec}) and input weight (w_{in}). Both weights are connecting to the hidden layer; however, at the initial step, the recurrent weight is 0.

RNNs have a profound weakness of being prone to both vanishing and exploding gradients problems. RNNs backpropagate the error calculated at the output through the entire length of the sequence. This entails backpropagating the error through all the hidden layers, which in turn involves repeatedly multiplying the error by the weights on the connections feeding activations from one hidden layer forward to the next hidden layer [40]. As a consequence, if the input sequence k is large, and total error weights are not equal to 1, the gradient will either decay to zero exponentially fast or grow exponentially fast.

One way to solve this problem is to use so-called Long Short-Term Memory networks (LSTMs). Figure 10 illustrates the overall LSTM stream at time step t-1, t and t+1. Meanwhile, Figure 11 shows the internal structure of an LSTM cell within an LSTM RNN network.

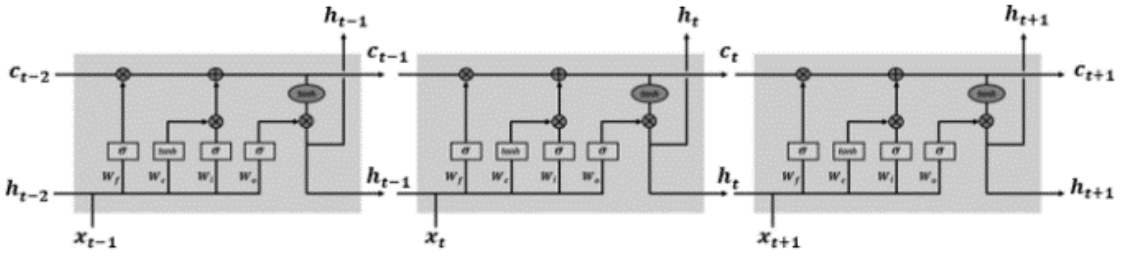


Figure 10: A LSTM network at time steps t-1, t and t+1 [26].

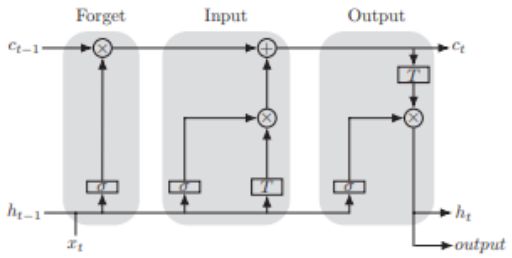


Figure 11: A closer look at an RNN cell within an RNN network at t-1 [26].

LSTMs mitigate the potential of vanishing and exploding gradient problems by introducing a three gates system: forget gate, input gate and output gate. The forget gate controls what

level of information in the cell state to discard; The input gate controls what new information will be encoded into the cell state; And, the output gate controls what information encoded in the cell state is sent to the network as input in the following time step. At the forget gate, the output can be summarised in the next mathematical term:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$$

At the input gate, the output comprehending two connected layers:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$\widetilde{c_t} = [\tanh\ activation]\,(W_c \cdot [h_{t-1}, x_t])$$

Therefore the output of the input gate can be rewritten as:

$$i_t \oplus \widetilde{c_t}$$

Lastly, the output gate has an activation in:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$$

Compare to the oridinary RNNs, LSTMs introduces a cell state vector:

$$c_t = \widetilde{c_t} \oplus f_t \oplus i_t \oplus c_{t-1} \quad \textit{where } c_{t-1} \textit{ is the state vector at t-1}$$

Essentially the gradient of the cell state vector in LSTMs is an addictive function made up of four terms ($\widetilde{c_t}, f_t, i_t \textit{ and } c_{t-1}$) instead of a multiplication function when backpropagating through an ordinary RNN. In the meantime, the presence of forgetting the gate's vector of activations allows the network to better control the gradient values at each time step using suitable parameter updates of the forget gates.

In addition to LSTMs, there is another famous RNN architecture called Gated Recurrent Unit (GRU). The GRU network is a lighter and faster alternative to the LSTMs. However, studies [24] [28] have shown that in NER tasks, LSTMs often lead to slightly better results compare to GRU networks. Thus in this study, the LSTMs is favoured over the GRU network.

3.9 Word Embedding and ELMo

Word embedding is the collective name for a set of language modelling and feature learning techniques in NLP where words or phrase from vocabulary are mapped to real-valued vectors in a predefined vector space. The purpose of word embedding is essentially to reduce the dimensionality of the input NLP data while learning underlying features or characteristics of the input NLP data [29]. The simplest form of a word embedding method is to denote each unique word or phrase in an NLP sequence to a real number then assign it to the next available location in a vector space. This method does not incorporate any advanced language modelling or feature learning techniques with merely representing words or phrases as vectors of real numbers. Hence no in-depth semantic features are extracted with these techniques. Therefore, there are a lot of sophisticated word embedding methods such as Word2Vec, FastText, GloVe, and WordRank [30] have been adopted by researchers and the industry in solving NLP tasks. These methods not only transforming words or phrases into vector representations but also uncovering additional semantic features. For our study, we proposed to incorporate a so-called Deep Contextualised word Representations (ELMo), which will be further explained in the later section.

3.10 State of the Art Deep Learning Architecture

The proposed deep learning models used in this project are largely derived from the bidirectional LSTM networks (BiLSTMs). In a regular LSTM, given a sequence of *N*-word tokens at different time steps ($t_1, t_2, \ldots, t_N$), the LSTM computes the probability of the sequence by modelling the probability of token t_k given the time steps ($t_1, t_2, \ldots, t_k$):

$$p(t_1, t_2 \ldots t_N) = \prod_{k=1}^{N} p\,(t_k | t_1, t_2 \ldots, t_{k-1})$$

Correlative to forwarding LSTM, a backward LSTM moving input in reverse order:

$$p(t_1, t_2 \dots t_N) = \prod_{k=1}^{N} p\,(t_k | t_{k+1}, t_{k+2} \dots, t_N)$$

Together, in a BiLSTM, the aim to maximise the log-likelihood that taking account of the log-likelihood at both forward and backward directions:

$$\sum_{k=1}^{N} (\log p\big(t_k | t_1, t_2 \dots, t_{k-1}; \theta_x, \overrightarrow{\theta_{LSTM}}, \theta_s\big) + \log p(t_k | t_{k+1}, t_{k+2} \dots, t_N; \theta_x, \overleftarrow{\theta_{LSTM}}, \theta_s))$$

where θ_x is the token vector representation and θ_s is the softmax activation layer.

3.10.1 BiLSTM – CRF

When using LSTMs alone in a NER task, the sequential word tokens or sentence are trained through both forward and backward LSTM. The output of LSTM is a sequence of probabilities for predicted NE labels. The dimension of the output sequence is depended on the length of the word sequence as well as the number of hidden cells in an LSTM. For example, if a sentence has 400 tokens and trained with an LSTM with 500 hidden cells, the output dimension will be 400 × 500. Eventually, a softmax classifier yields the predicted entity label for a token by finding the maximum probability among all possible NE labels, as indicated in Figure 12:

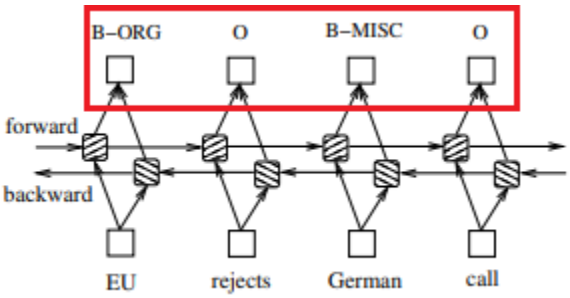


Figure 12: Diagram of BiLSTM in a NER task without a CRF classifier in the output layer [6].

Although BiLSTM unfolds the contextual features of a word token with respect to its adjacent neighbours, it does not incorporate the information of adjacent entity labels during the last classification stage [6]. It is possible to substitute the final softmax classifier with

a CRF classifier to take the neighbouring entity label information into account. Using CRF to process the output from a BiLSTM network is almost the same as using a standalone CRF in a NER task with a slight difference. When using the standalone version of CRF to predict NE labels, it finds the most likely NE label combinations for a sequence of word tokens by incorporating the adjacent NE label weight (ω). However, in BiLSTM – CRF, the weighting is applied to the output of the BiLSTM network, instead of directly onto the sequence of word tokens. Figure 13 and 14 shows an integrated solution of BiLSTM and CRF:

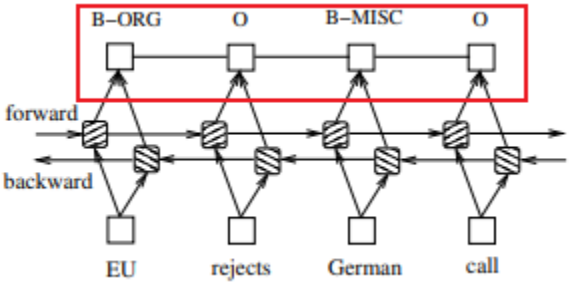


Figure:

Figure 13: Diagram of BiLSTM – CRF in a NER task with a CRF classifier in the output layer [6].

Algorithm 1 Bidirectional LSTM CRF model training procedure

```
for each epoch do
    for each batch do
        1) bidirectional LSTM-CRF model forward pass:
            forward pass for forward state LSTM
            forward pass for backward state LSTM
        2) CRF layer forward and backward pass
        3) bidirectional LSTM-CRF model backward pass:

            backward pass for forward state LSTM
            backward pass for backward state LSTM
        4) update parameters
    end for
end for
```

Figure:

Figure 14: Psuedo procedure for BiLSTM - CRF during training [6].

As mentioned in the softmax section earlier (section 3.6), CRF does not always guarantee a better result when compared to a softmax classifier, even with the additional neighbouring information [24]. Also, concretely, there are other ways, such as the

Maximum Entropy Markov models [31] can be used to decode neighbouring tagging information. Yet CRF is recognised to be the more robust one [6].

3.10.2 BiLSTM – CNN – CRF

BiLSTM – CNN – CRF is considering as an advancement of the aforementioned BiLSTM – CRF architecture, that not only decoding the neighbouring prediction outcomes at the word token level but also extracting features at character level within each word token of a sequence. For instance, given a sample sentence 'We are playing soccer', a typical BiLSTM – CNN – CRF model can be illustrated as in Figure 15 and 16 below:

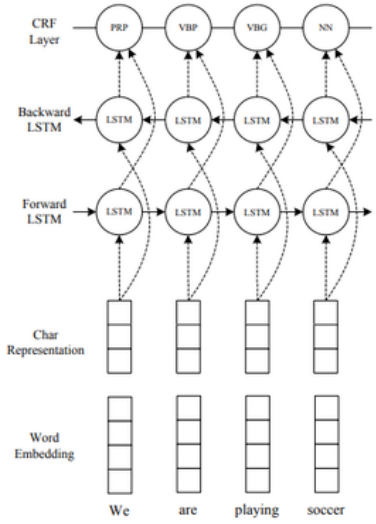


*Figure 15: The (unrolled) BiLSTM – CNN - CRF architecture for a NER task. *sample sentence 'We are playing soccer' [8].*

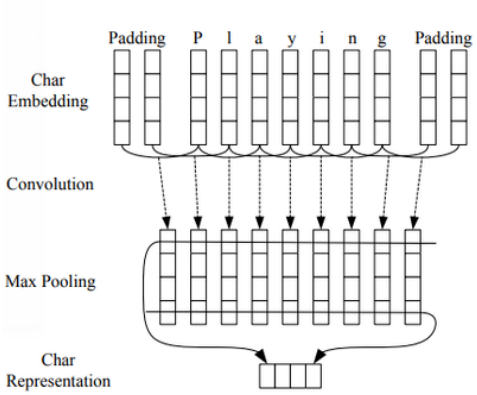


Figure 16: A closer look at the CNN component at the token 'Playing' from the Figure above (Figure15) [8].

From Figure 15 and Figure 16 above, the feature extraction at the word token level for BiLSTM-CNN-CRF resembles the BiLSTM-CRF. But, a 1-dimension CNN with max-pooling is used to extract character features for each token or work found in a sentence. Then the output vector from CNN is concatenated together with the output of BiLSTM before sending into the final CRF classifier.

Despite both BiLSTM-CRF and BiLSTM-CNN-CRF have shown success in resolving many NER challenges [6] [7] [8], they are usually tested with structured NLP datasets with innate grammatical characteristics, as examples, news, articles or document from English literature. Unlike these datasets, documents such as business invoices commonly constructed with keywords or entity indicators such as 'Amount:' or 'Date:' and does not necessarily follow any grammars. On top that, when aiming to extract a handful NE from an unstructured document, a large proportion of word tokens from the document are irrelevant. This leads to a relatively sparse NE label set with most NE labelled as irrelevant (in BIO tagging, the 'O' annotation). This posts a question of whether considering the neighbouring NE weights still worth doing.

3.10.3 ELMo - BiLSTM

One of the state-of-art word embedding [32] that is gaining phenomenal popularity in the field of NLP is the ELMo word embedding. ELMo, shorthand for Embeddings from Language Model (LM), is a task-specific and contextual based word embedding method. During training, the ELMo embedding works similar to the BiLSTM – CNN mentioned earlier (section 3.10.2), it decomposes a token sequence down to character level embeddings using a CNN network followed by a max-pooling. However, ELMo then processes these character-level word representations using a two-layer RNN network to gain a contextual awareness according to the current token sequence in which characters and tokens are located, hence task-specific. When combining the ELMo embedding on top of an additional BiLSTM network, the original output sequence for BiLSTM $\{x_k, \overrightarrow{h_{kj}}, \overleftarrow{h_{kj}} \mid j = 1,2, \dots, L\}$ (where j is the hidden cells defined in an LSTM network, x_k is the token sequence) is now $\{\gamma^{task} \times (s_0^{task} \times x_k, s_1^{task} \times \overrightarrow{h_{kj}}, s_2^{task} \times \overleftarrow{h_{kj}}) \mid j =$

$1,2,\ldots,L\}$ (where γ^{task} represents the task-specific scaling while s_i represents a softmax-normalised weight on the hidden representations from the ELMo language model). There are no other major alterations to the fundamental setup of the BiLSTM. The final output sequence is eventually processed with a softmax classifier (Figure 17).

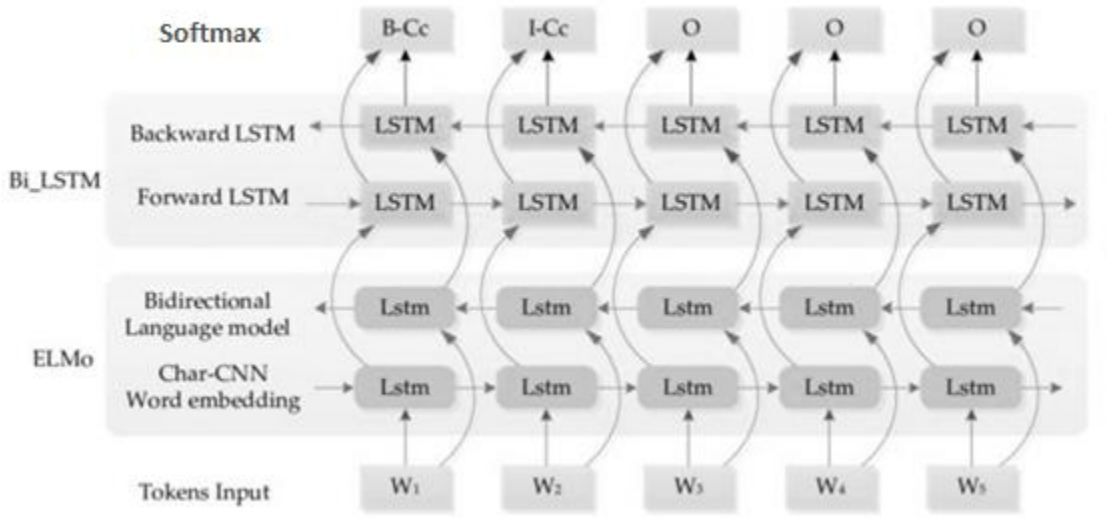


*Figure 17: ELMo - BiLSTM with a softmax classifier at the output layer *inspired by [9].*

Compare to other word embedding approaches, ELMo word representations are functions of the entire input sentence and capable of extracting the hidden state of each layer for the input sequence of words or phrases. Pretrained ELMo model weights based on the 1 Billion Work Benchmark are freely available and is used in this study.

Using ELMo on top of BiLSTM network is a relatively fresh approach, previous work [32] are largely evaluated based on structured NLP data. Despite the fact that it has shown some success in NE task against unstructured biomedical data [9], its capacity of extracting NE information from commercial receipts is not widely tested.

It is worth to mention, as ELMo-BiLSTM proposed in [32] uses a softmax activation for the classification task. However, given the literature provided earlier as in BiLSTM-CRF and BiLSTM-CNN-CRF (section 3.10.1 and 3.10.2), it is not surprising that ELMo – BiLSTM can be altered to ELMo-BiLSTM-CRF with a CRF classifier instead of a softmax

activation. This way, using CRF as the classifier may improve the ELMo-BiLSTM result as CRF incorporate adjacent label weights. But this hypothesis remains untested.

4

Methodology

Below is a flowchart that shows the abstract workflow of building, training and evaluating the NER deep learning models used in this project (Figure 18):

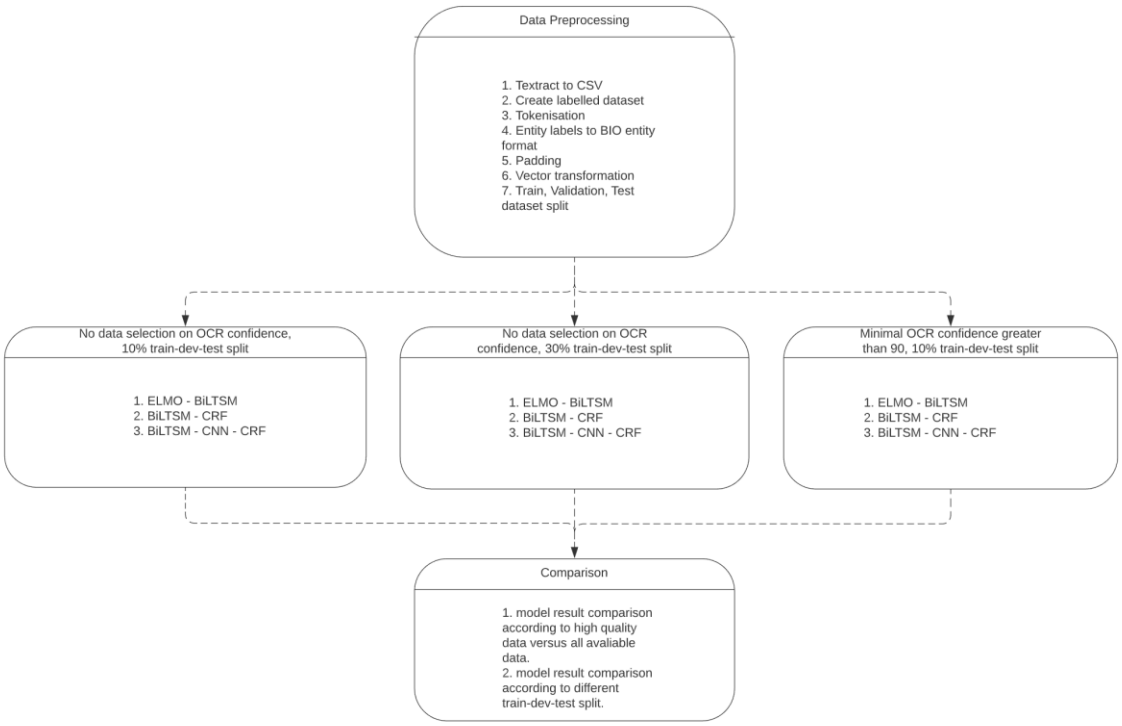


Figure 18: Abstract training workflow of this study.

Unlike a standard NLP process which assumes the underlying data inherit grammatical rules, most texts found on business invoices are unstructured. Therefore, earlier mentioned NLP data preprocessing methods (section 3.3) such as lemmatisation, stemming, and Part-of-speech tagging, are not implemented in this project.

This section will provide detailed information on the methodology applied in this project. Such information includes the hardware and software used to achieve the goal of this project; background details on data format and critical steps in data preprocessing; the architecture of NER deep learning models, and lastly the methods used to evaluate the deep learning models.

4.1 Development Environment

The coding implementation of our deep learning models is primarily done in Python 3.7.1 64bit. Core Python packages used in this project that are not part of the out-of-box Python installation include spaCy v2.3, pandas v1.0.3, matplotlib v3.1.3, scikit-learn 0.23, TensorFlow v1.15, Keras v2.3.1 and bokeh v2.0.2.

Models are trained on a laptop with 32GB of RAM, 1TB SSD hard drive, Intel Core i7-8750H CPU at 2.20 GHz and an external Nvidia GTX2070 8GB graphic card (GPU) at 1620 MHz.

The GPU was running with Nvidia 456.71 driver and configured with CUDA 10.0.130 in order to train the deep learning model with GPU instead of CPU.

4.2 Data and data preprocessing

Overall, a total of 1281 invoice and GST receipts were used in this project with a 5.04 MB disk size. On average, the OCR reading confidence score is around 92.63 per document, which indicates a good quality OCR conversion from image to text. The OCR quality for most records is over 90 (Figure 19).

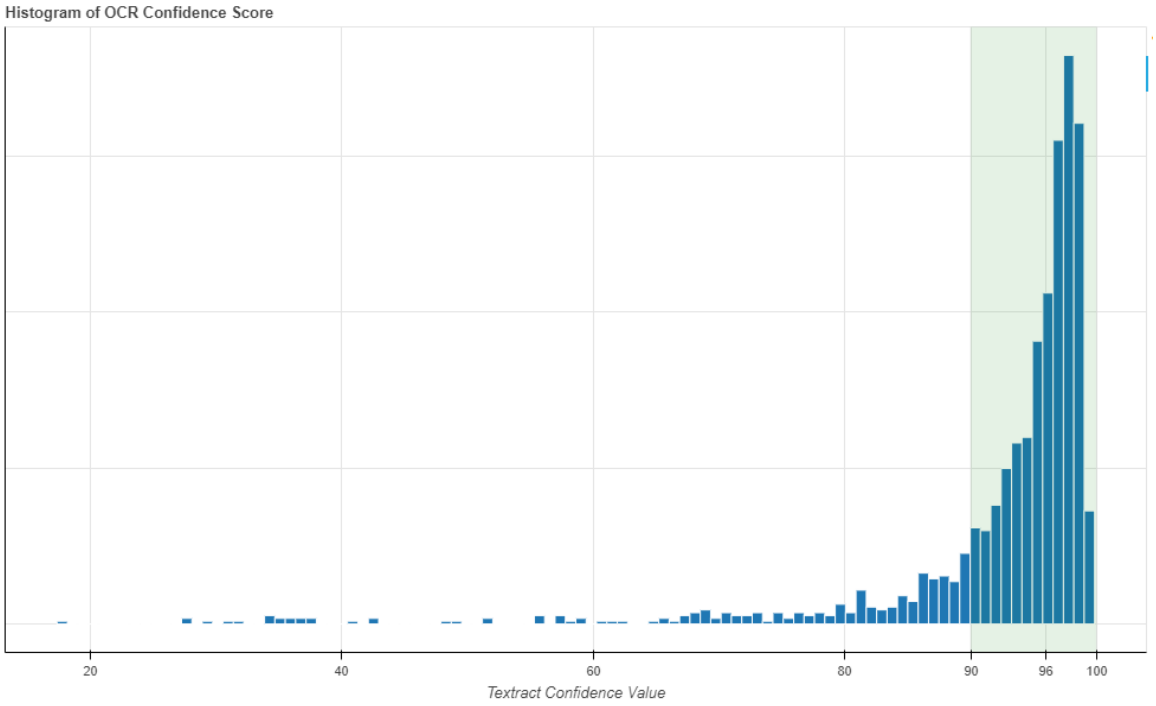


Figure 19: Histogram of the confidence score per document (the confidence score per document is calculated by averaging the confidence score for each text elements (i.e. words) found in a document)*

However, there are few invoices poorly processed by Amazon Textract, with a combined confidence score of all text elements as low as 17.38. The OCR readings on these entries are extremely uninterpretable. In contrast, there is a total of 1098 receipts have an above 90 overall confidence score per document with a 4.11 MB disk size.

4.2.1 Data extraction

As mentioned in the background section, the raw data used in this project is the output from Amazon Textract OCR service in JSON format. The default Textract output JSON file contains a large amount of information such as 'item id' and 'geometry' that is irrelevant to the overall NER goal. Thus, only text that stored at the Textract's 'Line' nodes is extracted and then saved into a CSV file for the later process, while irrelevant information is discarded.

In order to explore the impact of OCR quality on the model performance, two sets of data are created: one makes no selection based on the per-document confidence score, while the other only contains invoice data have achieved more than 90 per-document confidence

score. This selection does not take into account of confidence score per individual text elements found on each invoice document, rather than a summarised confidence score for every single receipt document hence 'per-document'.

4.2.2 Labelling

In this study, we focused on a few entities: 'Merchant', 'GST', 'Date', 'Price' and 'Time'. In order to carry out the NER process, the extracted text data is initially annotated with targeted NE categories ('merchant', 'amount', 'GST', 'date' and 'time'). A combination of manual labelling, regular expression matching and Python spaCy's phrase matching is used to label a part of the text in each receipt text with the NE categories by stating the NE category, and the location of the NE is found in a receipt using start and end indices (Table 1)

TEXT	NE LABEL
GOURMET BURGER KITCHEN, 45 GREENWICH CHURCH, ST, LONDON, M*07469, TID, TID*1*9345, 9345, AID A0000000031010, VISA, VISA, E 30 3, 6162, ICC, PAN.SEQ 01, SALE, CARDHOLDER COPY, PLEASE KEEP THIS RECEIPT, FOR YOUR RECORDS, AMOUNT, E17.75, VERIFIED BY SIGNATURE, THANK YOU, 14:28 22/11/14	{'entities': [(275, 283, 'DATE'), (269, 274, 'TIME'), (0, 22, 'MERCHANT')]}

Table 1: Example of NE label for a sample invoice data entry (inside the parenthesis, the number indicates the start and end index of an entity found with respect to the total length of a business invoice, the label indicates the NE category).*

Moreover, because the proposed deep learning models are expecting the data is labelled in BIO tagging system, the NE label is converted to BIO tags after the original text is tokenised (Table 2). All text tokens that do not belong to a NE category in an invoice receipt, including punctuations, symbols, digits, are simply labelled with the 'outside' (O) tag. All text tokens that belong to a NE category on an invoice, including punctions, symbols, digits are labelled with corresponding BIO NE tags. There are total 12 BIO NE labels used in this project, including one for the padding token (B-MERCHANT, I-MERCHANT, B-DATE, I-DATE, B-AMOUNT, I-AMOUNT, B-GST, I-GST, B-TIME, I-TIME, O and PADDING)

invoice_id	token	tag

invoice_1	gourmet	B-MERCHANT
invoice_1	burger	I-MERCHANT
invoice_1	kitchen	I-MERCHANT
invoice_1	,	O
invoice_1	45	O
invoice_1	greenwich	O
invoice_1	church	O
invoice_1	,	O
invoice_1	st	O
invoice_1	,	O
……	…....	……

Table 2: Example of BIO annotation on a sample invoice data entry.

Within the original dataset, the largest proportion of NE is the 'amount' entity (42% of total), whereas the 'time' entity only accounts for roughly 8% of the total NE types (Table 3).

B-AMOUNT	21.45%
I-AMOUNT	20.66%
B-MERCHANT	9.96%
I-MERCHANT	11.02%
B-DATE	10.07%
I-DATE	9.31%
B-GST	1.92%
I-GST	7.75%
B-TIME	6.14%
I-TIME	1.71%

*Table 3: NE label distribution within the original dataset (*tagging is in BIO format)*

In the dataset with more than 90 per-document confidence score, the proportion of NE types is roughly the same. However, there is a slight decrease in the 'amount' entity while a slight increase in the 'merchant' entity (Table 4).

B-AMOUNT	20.80%
I-AMOUNT	20.04%
B-MERCHANT	10.18%
I-MERCHANT	11.20%
B-DATE	10.14%
I-DATE	9.75%
B-GST	1.95%
I-GST	7.86%
B-TIME	6.25%
I-TIME	1.84%

*Table 4: NE label distribution within the original dataset (*tagging is in BIO format).*

4.2.3 Tokenisation

The tokenisation is primarily accomplished by using the spaCy's default English tokeniser [33]. After the tokenisation process, all text a normalised to lower case to reduce the computing complexity. The longest invoice contains 496 tokens, and the longest token contains 63 characters. Also, on average, there are 159 tokens per business invoice (Figure 20).

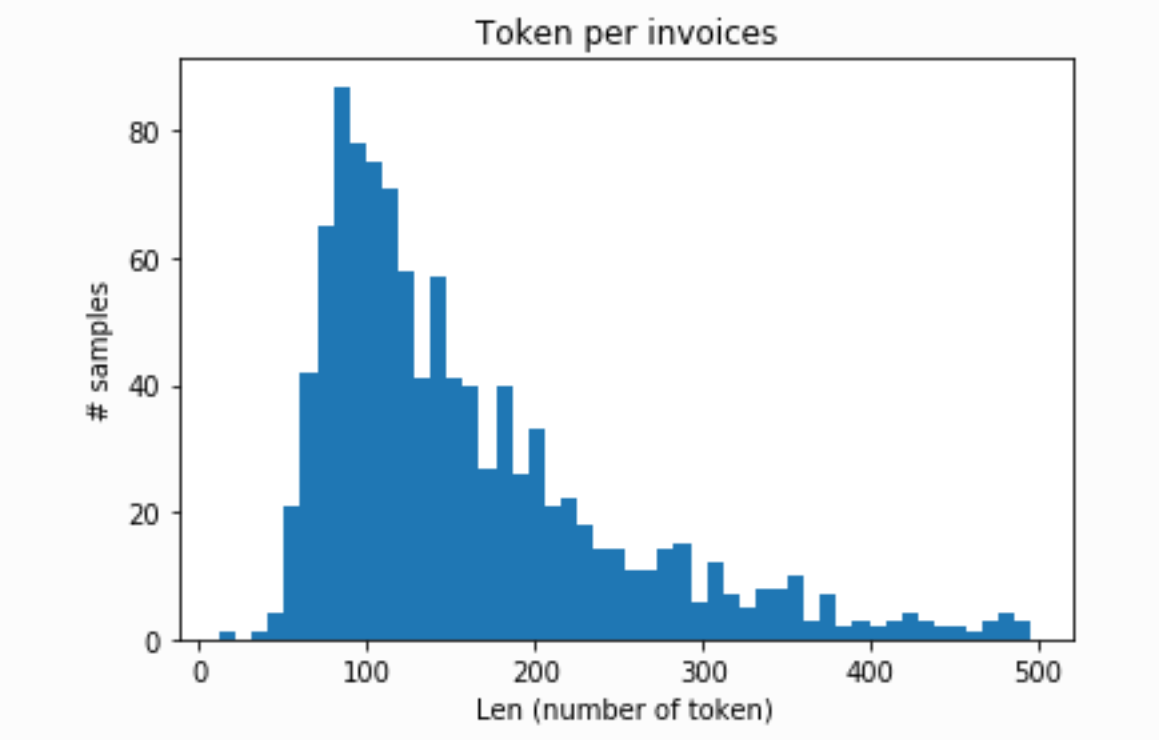


Figure 20: Histogram of token length for all invoices, regardless of their confidence score.

4.2.4 Padding

Prepadding strategy is used as it has been suggested showing better result compare to post padding [34]. The maximum input sequence length for the LSTM network is set to the longest token sequence found across all business invoice data.

Since the longest tax invoice contains 496 tokens, tax invoices have fewer than 496

tokens will be padded with 'Pad Token' until they all achieve the maximum token length. During the training, pad tokens will be labelled additionally as 'Padding' instead of 'O'.

4.2.5 Transforming into vectors

NE labels are mapped to unique integers to further reduce the computing complexity and resource consumption as in Figure 21:

```
{1: 'I-MERCHANT',
 2: 'B-DATE',
 3: 'B-TIME',
 4: 'I-TIME',
 5: 'I-DATE',
 6: 'B-MERCHANT',
 7: 'O',
 8: 'B-GST',
 9: 'B-AMOUNT',
 10: 'I-AMOUNT',
 11: 'I-GST',
 0: 'PADtoken'}
```

Figure 21: BIO tags to vector indices.

For BiLSTM, the lexical input tokens need to be converted into vector format as well. For the BiLSTM-CRF and BiLSTM -CNN-CRF, word embedding is simply done by allocating tokens onto the next available room in a vector space. Because characters-level training also occurs in BiLSTM -CNN-CRF, character-level embedding is also carried out for BiLSTM -CNN-CRF in a comparable fashion as the word-level embedding. In contrast, the ELMo based BiLSTM uses the ELMo contextualised word embedding to create vector representation for the input tokens.

4.2.6 Pre-selection based on the confidence score

In this project, model performance and training results are compared when training using data with an overall 90 confidence score versus training using all available data. Therefore a set of 'high quality' (condifence score >90) data is created.

4.2.7 Training, Validation and Testing split

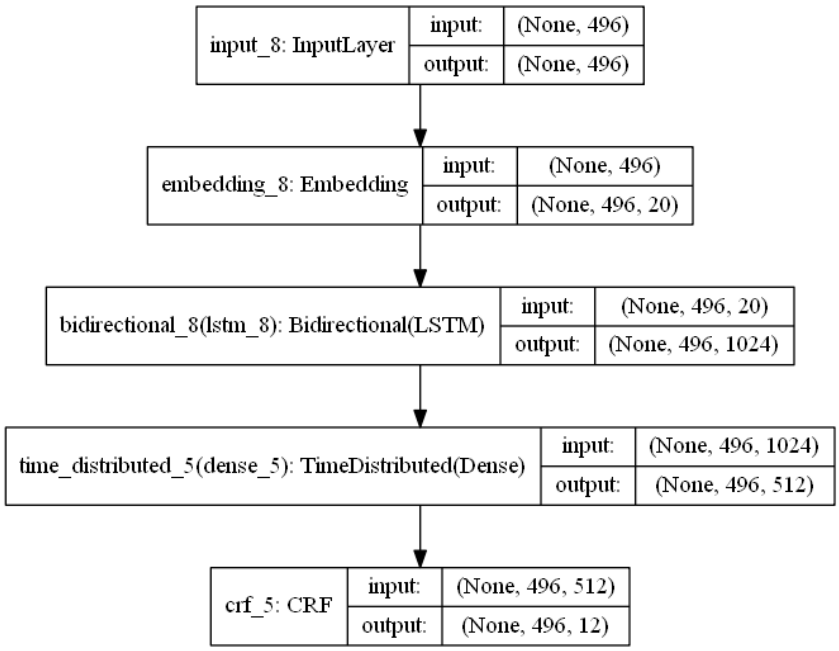
Both of the full datasets and selected datasets based on >90 per-document confidence score are split into training, validation and testing set. There are two split proportion schemes used: one retains 81% training set, 9% validation set and 10% testing set while the other retains 49% training set, 21% validation set and 30% testing set. In the first split scheme, the testing set is in 1:9 ratio compared to the training set plus the validation set, and the validation set is in 1:9 ratio compare to the training set. In the second split scheme, the testing set is in 3:7 ratio compared to the training set plus the validation set, and the validation set in 3:7 ratio compare to the training set.

4.3 Model implementation

This section outlines the implementation of all three state-of-art deep learning models that used to achieve the NER goal for this project. Due to the limited scope, hyperparameter tuning and the evaluation on the impact of different hyperparameter settings on model performance are not examined in-depth in this project. The selection of hyperparameters was based mainly on the work carried out in [35]. In order to make a cross-architecture comparison, all LSTM neural network is configured with 512 *Recurrent Units* across the three deep learning architectures with a 10% *Recurrent Dropout rate*. The *Adam* optimiser is chosen to be the default optimisation algorithm in neural networks. All three deep learning models are trained with a *Batch Size* of 8, and their training *weights* are evaluated based on *validation accuracy* during training.

4.3.1 BiLSTM - CRF

The graphic representation of BiLSTM – CRF architecture implemented in Tensorflow is shown below:



Figure

Figure 22: Graphic representation of BiLSTM implementation in Tensorflow.

The input sequence has a fixed length of 496 tokens. Before feeding into the BiLSTM network, the sequence is translated into a vector with a 496 × 30 dimension. As mentioned at the beginning of this section, the BiLSTM has a total of 1024 recurrent units of 512 units for each LSTM. Before the final classification using CRF classifier, the output of BiLSTM is mapped with a *Dense* layer with *Relu* activation to reduce the dimension to a single LSTM network (512 units) and then a *TimeDistributed* layer to keep a one-to-one relation on input and output.

The final output is a vector with 496 × 12 dimensions as 496 is maximum sequence length and 12 is the total number of NE labels.

4.3.2 BiLSTM - CNN-CRF

The graphic representation of BiLSTM – CNN – CRF architecture implemented in Tensorflow is shown below:

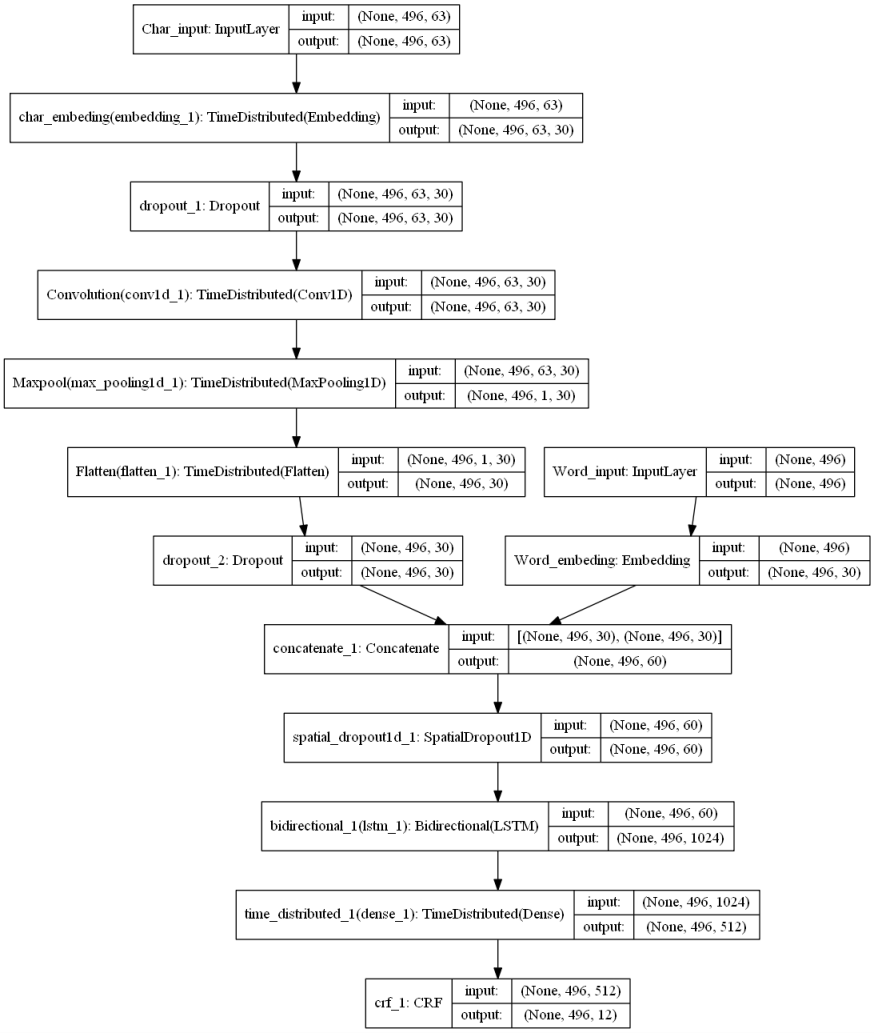


Figure 23: Graphic representation of BiLSTM – CNN – CRF implementation in Tensorflow.

The BiLSTM – CNN – CRF is analogous to BiLSTM – CRF with an additional CNN component. The BiLSTM – CNN – CRF is configured with a 1-dimension CNN network to encode character-level information of a word token into its character-level representation as suggested in the literature [31]. The 1-dimension CNN neural network has a *kernel size* of 3 and *filter* number of 30. The input dimension for the CNN network is 496 × 63, as there are maximum 496 tokens in the longest sequence and the longest token has 63 characters within the entire dataset. After the characters are transformed into vector space, a 10% *dropout rate* is applied. Similar to the BiLSTM – CRF architecture, the embedding dimension is 30 (for both characters and words), and the activation function is *Relu* (for the *Dense* layer and 1-dimension CNN). The 1-dimension CNN is configured with a *kernel size* of 5, 30 *filters*, filter window is moving 1 *stride* at a time and the same *padding* as the

input. Before the final classification stag with a CRF classifier, word-level embedding and character level embedding are concatenated together with a 20% *Spatial Dropout rate* applied. All other hyperparameter settings, feature input and output, are the same for both BiLSTM – CNN – CRF and BiLSTM – CRF.

4.3.3 ELMo – BiLSTM

The graphic representation of ELMo – BiLSTM architecture implemented in Tensorflow is shown below:

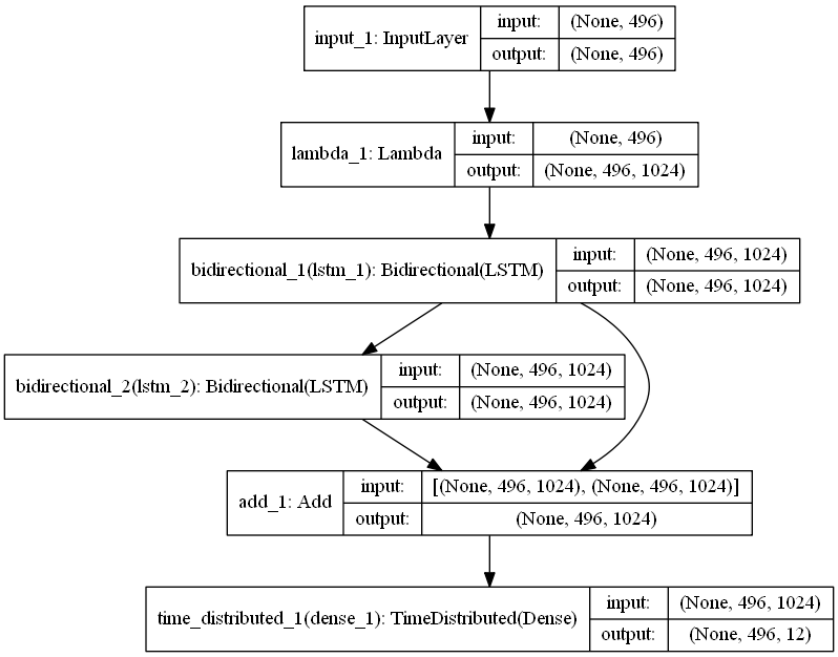


Figure 24: Graphic representation of ELMo – BiLSTM implementation in TensorFlow.

In ELMo – BiLSTM, two bidirectional LSTM layers are used to train the ELMo embedded tokens. They are linked with *residual connections*, as stated in [32]. The hyperparameters used in ELMo – BiLSTM is mostly comparable to the BiLSTM – CRF model with two exceptions: first, additional 10% standard *dropout rate* is applied on top of the 10% *recurrent dropout rate* to prevent overfitting further; secondly as ELMo uses softmax as its classifier, the model is set to minimise the *sparse categorical cross-entropy* loss function instead of the CRF loss function.

5

Result and Discussion

5.1 Results

5.1.1 Summary

In summary, all three models retain a good overall accuracy and learning rate. The ELMo – BiLSTM is the best performing one. It is less affected when there is a change in the underlying data structure and quality. Its learning ability and performance are the most robust and reliable across all three models, even when the training data contain OCR noises such as misreading. Across all three models, they all have difficulties when predicting the correct merchant NE type. Besides, the predictability for the 'GST' NE varies the most when using BiLSTM – CRF and BiLSTM – CNN – CRF with the different train/validation/test data structure.

5.1.2 Model Computational Cost

Training with LSTM – CRF costs the least GPU memory consumption: 3.6 GB with full dataset, 3.2 GB with data that have high confidence score and the least time to run: around 6 hours with the dataset. With an additional CNN layer, the LSTM – CNN – CRF costs slightly more memory and time to complete: 4.1 GB GPU memory with full dataset, 3.8 GB GPU memory with data that have high confidence score, and takes around 8 hours to train with both data split schemes. ELMo – BiLSTM is the most computational expensive one amongst all three models: it consumes about 7.2 GB GPU memory with the full dataset, 6.8 GB GPU memory with the high-quality dataset, and it takes approximately 12 – 14 hours to complete the training.

5.1.3 Model evaluation

The performance of our models is primarily evaluated on the confusion matrices of *Precision*, *Recall* and *F1-score* for each NE types (merchant, amount, date, GST and time) as well as the validation accuracy and loss graph during training. The standard *residual mean squared error* is not used as we encountered an imbalanced NE label distribution within our data.

$$Precision = \frac{True\ Positive}{True\ Postive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Postive + False\ Negative}$$

$$F1 - Score = \frac{2 \times True\ Positive}{2 \times True\ Postive + False\ Negative + False\ Positive}$$

5.1.3.1 Trained with all available data (split 9:1 between (train + validation) and test set, then 9:1 between train and validation set)

	precision	recall	f1-score	support
AMOUNT	0.88	0.88	0.88	391
DATE	0.61	0.30	0.40	183
GST	0.47	0.51	0.49	35
MERCHANT	0.52	0.41	0.46	194
TIME	0.80	0.37	0.50	123

(a)

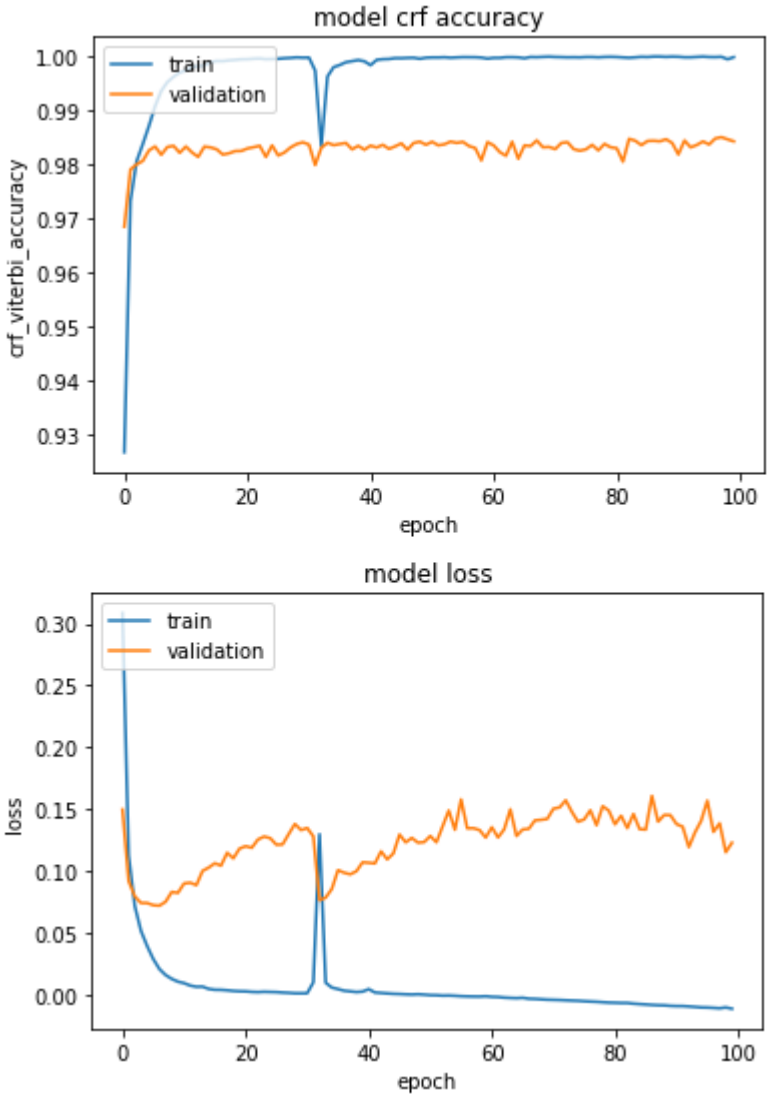
	precision	recall	f1-score	support
AMOUNT	0.93	0.90	0.92	476
DATE	0.61	0.38	0.47	215
GST	0.38	0.55	0.45	38
MERCHANT	0.55	0.45	0.50	223
TIME	0.93	0.40	0.56	135

(b)

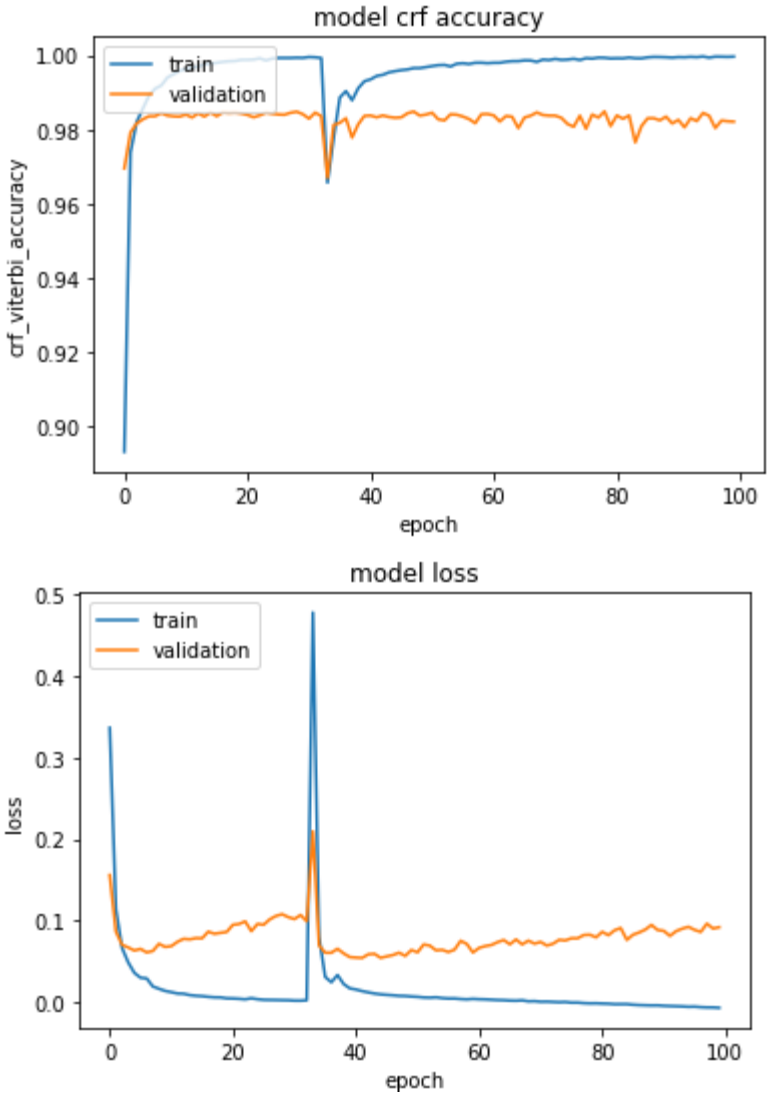
	precision	recall	f1-score	support
AMOUNT	0.94	0.95	0.95	390
DATE	0.84	0.88	0.86	180
GST	0.80	0.99	0.89	35
MERCHANT	0.60	0.54	0.57	193
TIME	0.96	0.98	0.97	121

(c)

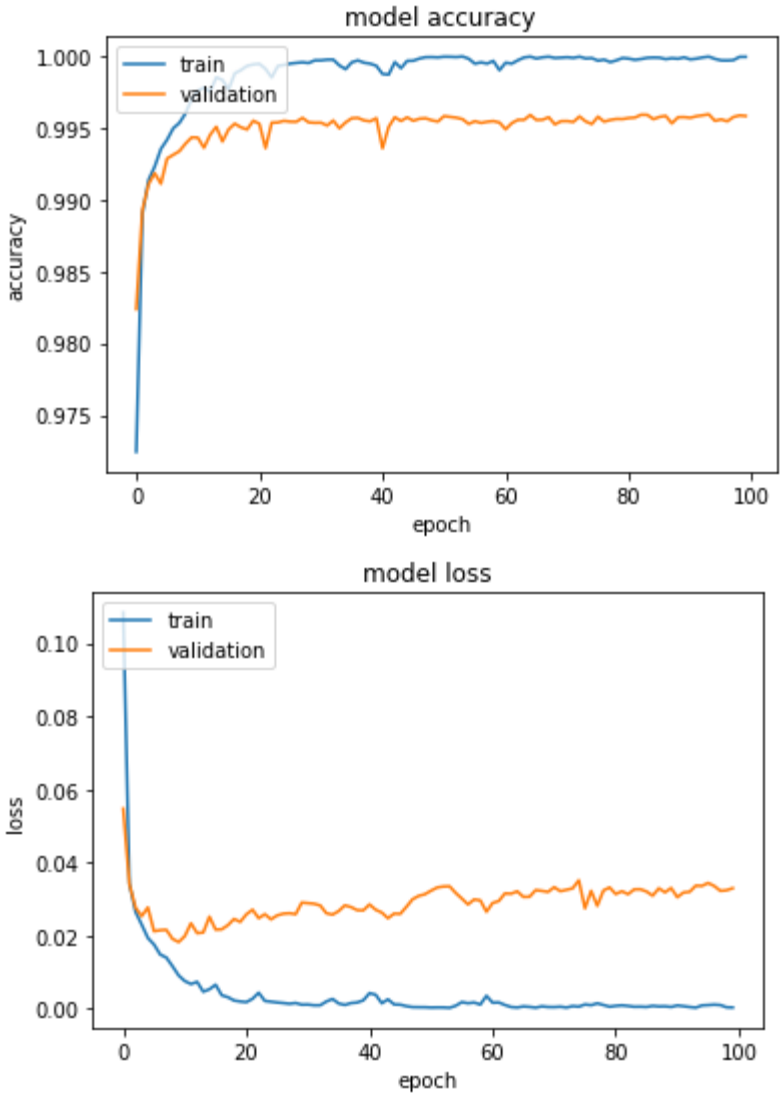
*Figure 25: NE classification report on all three different deep learning architectures. *a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*



(a)


model crf accuracy
train
validation
crf_viterbi_accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0
20
40
60
80
100
epoch
model loss
train
validation
loss
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
epoch

(b)



(c)

*Figure 26: Cross-validation accuracy and loss graph during training. *trained with 100 epochs, a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*

When training with all available data in a (9:1):1 split among training/validation/test split, validation accuracy and loss tend to become stable after around 45 epochs iterations for all three models (Figure 26). According to the validation accuracy graph (Figure 26), we observed a relatively strong pattern of overfitting when training using the LSTM – CRF model compare to the others. Meanwhile, according to the loss graph, all three models retained a good learning rate. Furthermore, based on the confusion matrices (Figure 25), there is a gradual improvement on the prediction result from the BiLSTM-CRF model to the BiLSTM – CNN – CRF model, then to the ELMo – BiLSTM model. Nevertheless, neither BiLSTM – CRF and BiLSTM – CNN – CRF made reliable prediction against the

'GST' NE tokens while the BiLSTM – CNN – CRF is the worst performing one overall. For the ELMo – BiLSTM model, it makes the worst prediction against the 'merchant' entity tokens in the same scenario.

5.3.1.2 Trained with all available data (split 7:3 between (train + validation) and test set, then 7:3 between train and validation set)

	precision	recall	f1-score	support
AMOUNT	0.92	0.88	0.90	1330
DATE	0.47	0.27	0.34	562
GST	0.50	0.56	0.53	117
MERCHANT	0.53	0.41	0.47	613
TIME	0.59	0.23	0.33	360

(a)

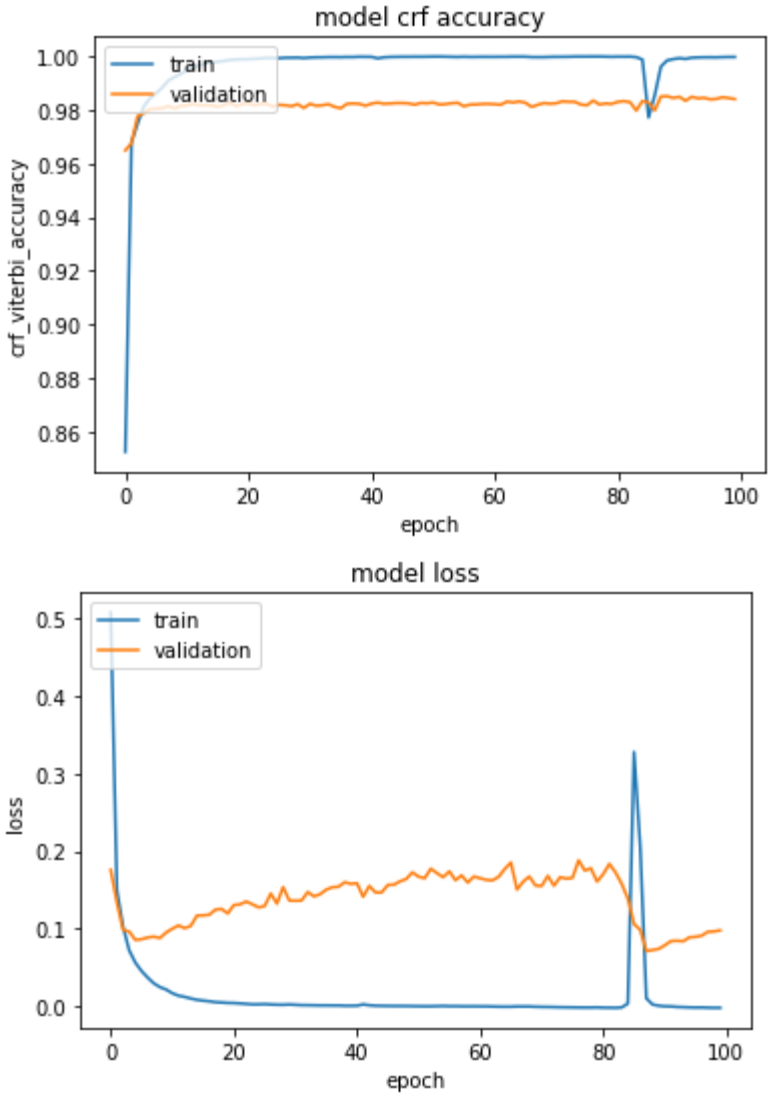
	precision	recall	f1-score	support
AMOUNT	0.91	0.87	0.89	1330
DATE	0.56	0.26	0.36	562
GST	0.43	0.50	0.46	117
MERCHANT	0.50	0.38	0.43	613
TIME	0.79	0.33	0.47	360

(b)

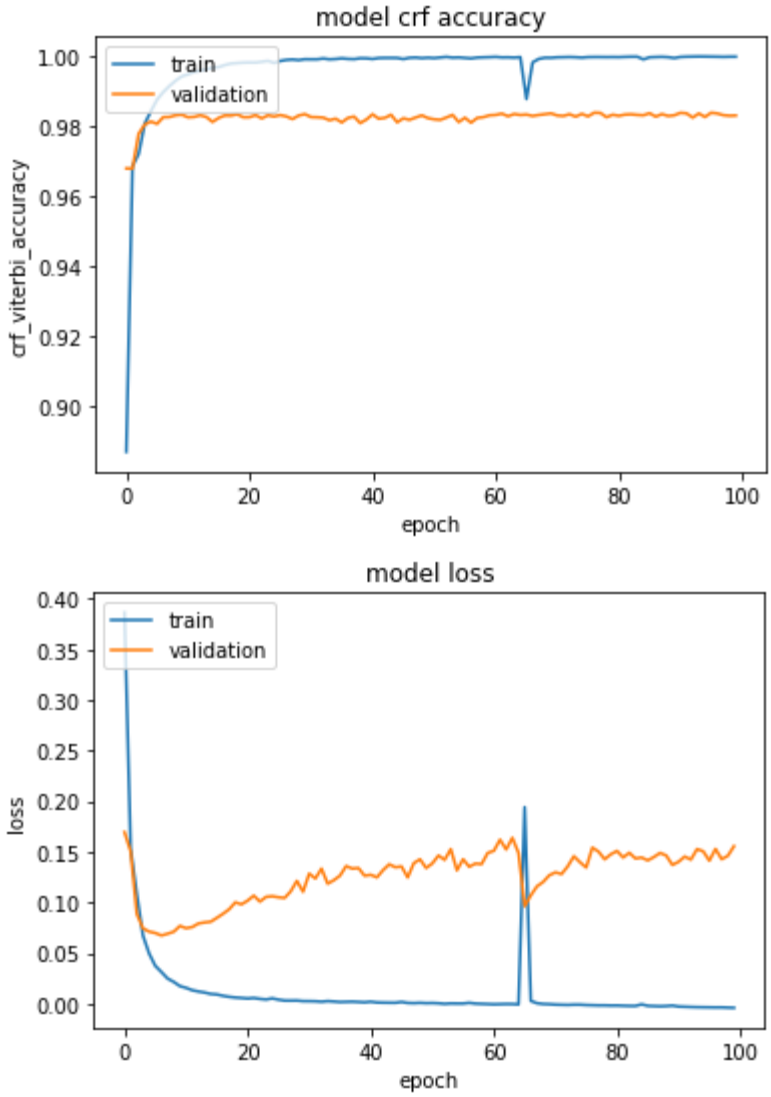
	precision	recall	f1-score	support
AMOUNT	0.95	0.92	0.93	1329
DATE	0.85	0.76	0.80	561
GST	0.84	0.95	0.89	117
MERCHANT	0.57	0.58	0.58	612
TIME	0.97	0.97	0.97	359

(c)

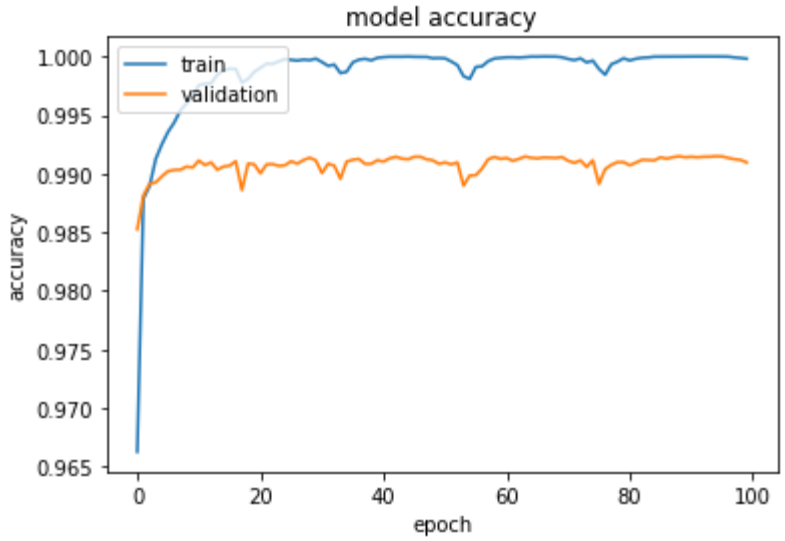
*Figure 27: NE classification report when training with (7:3):3 ratios among train, validation and test set. *a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*

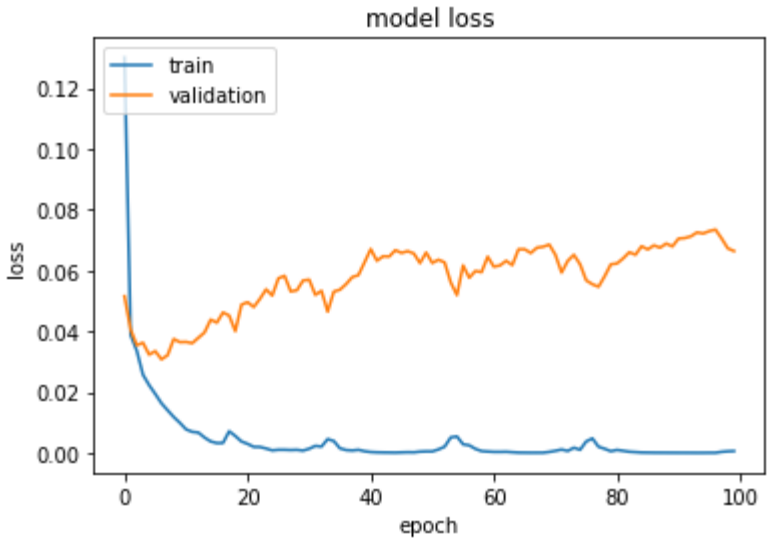

model crf accuracy
train
validation
crf_viterbi_accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
20
40
60
80
100
epoch
model loss
train
validation
loss
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
epoch

(a)


model crf accuracy
train
validation
crf_viterbi_accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0
20
40
60
80
100
epoch
model loss
train
validation
loss
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
epoch

(b)





(c)

*Figure 28: Cross-validation accuracy and loss graph during training with (7:3):3 ratio among train, validation and test set. *trained with 100 epochs, a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*

In this training session, we split the data in a 7:3 ration between train plus validation set and test set, then a 7:3 between the train and validation set. According to Figure 27 and Figure 28 above with comparison to the first training strategy (section 5.3.1.2), this time, we observed an overfitting in both of the BiLSTM – CRF and BiLSTM – CNN – CRF. We also saw a decrease in the prediction performance for both of the BiLSTM – CRF and BiLSTM – CNN – CRF model. However, with less training but more validation dataset, BiLSTM – CNN – CRF produces better prediction results against the 'GST' NE. Meanwhile, for the ELMo – BiLSTM model, despite having a slightly worsened results when predicting

the 'merchant' NE, its overall performance reasonably resembles what has been seen in the earlier figures (Figure 25 and Figure 26). We also see a slight decrease in the overfitting for the ELMo – BiLSTM model. Overall, the ELMo – BiLSTM is still the best performing one amongst all three models.

5.3.1.3 Trained with data with 90 overall confidence score per invoice (split 9:1 between (train + validation) and test set, then 9:1 between train and validation set)

	precision	recall	f1-score	support
AMOUNT	0.89	0.84	0.86	377
DATE	0.68	0.32	0.44	186
GST	0.38	0.48	0.42	31
MERCHANT	0.55	0.50	0.52	150
TIME	0.91	0.44	0.59	112

(a)

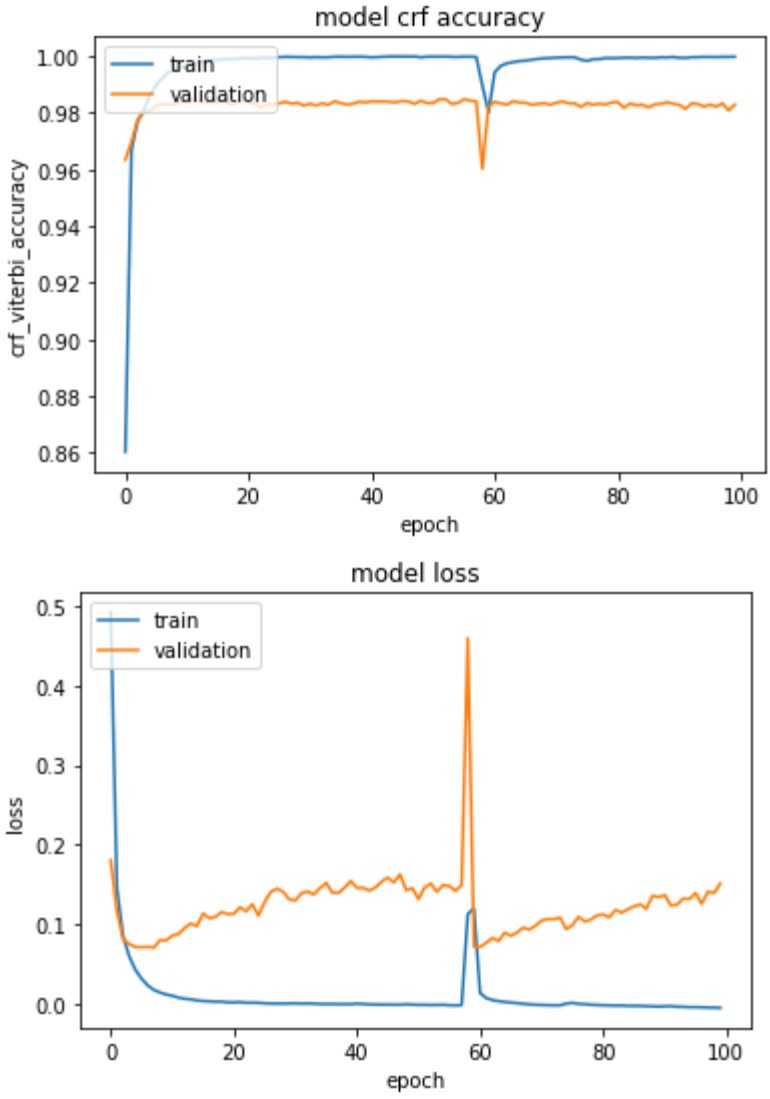
	precision	recall	f1-score	support
AMOUNT	0.94	0.84	0.89	377
DATE	0.78	0.32	0.45	186
GST	0.68	0.61	0.64	31
MERCHANT	0.57	0.48	0.52	150
TIME	0.86	0.45	0.59	112

(b)

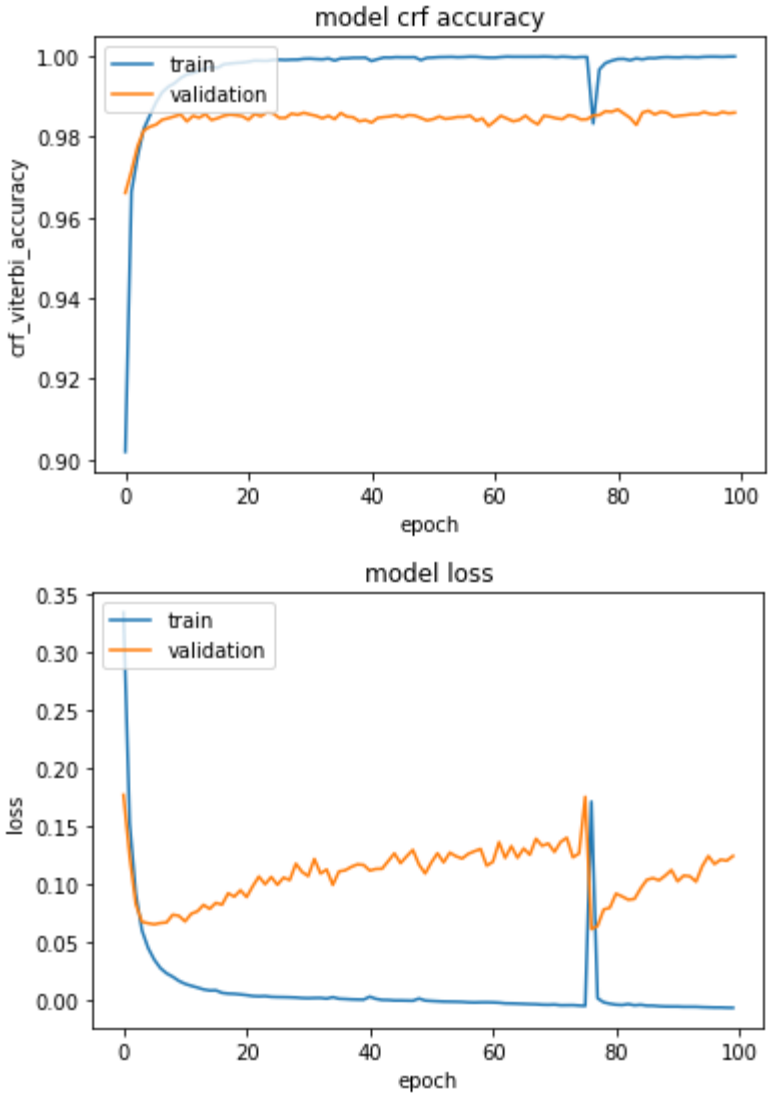
	precision	recall	f1-score	support
AMOUNT	0.95	0.93	0.94	377
DATE	0.87	0.91	0.89	184
GST	0.81	0.97	0.88	31
MERCHANT	0.61	0.48	0.54	149
TIME	0.96	0.99	0.97	110

(c)

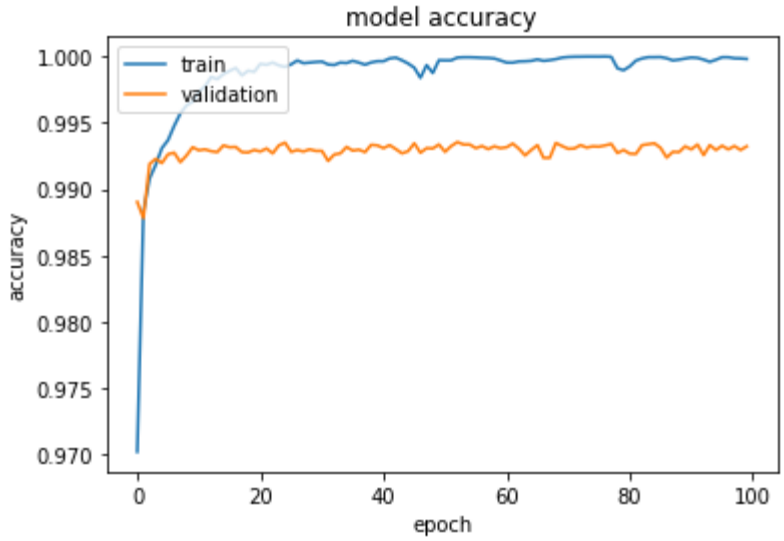
*Figure 29: NE classification report when using data achieve a minimum 90 confidence score per invoice. *a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*

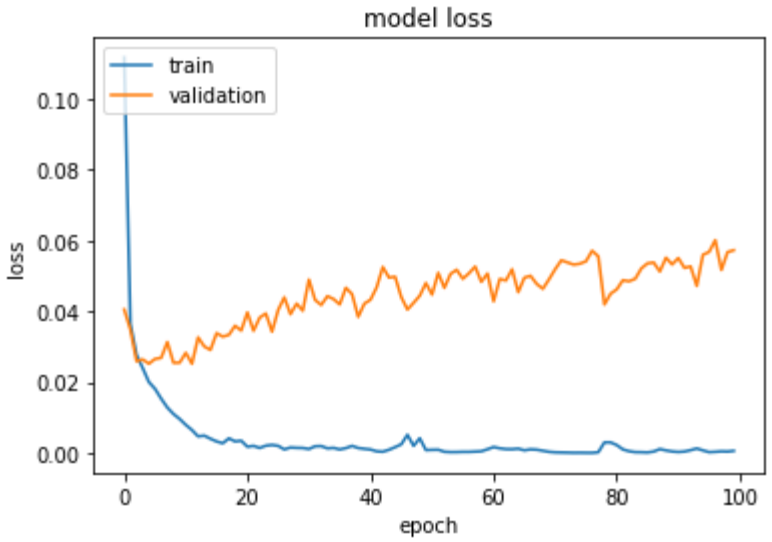

model crf accuracy
train
validation
crf_viterbi_accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
20
40
60
80
100
epoch
model loss
train
validation
loss
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
epoch

(a)


model crf accuracy
train
validation
crf_viterbi_accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0
20
40
60
80
100
epoch
model loss
train
validation
loss
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
epoch

(b)





(c)

*Figure 30: Cross-validation accuracy and loss graph during training with data achieve a minimum 90 confidence score per invoice. *trained with 100 epochs, a. BiLSTM-CRF; b* BiLSTM-CNN-CRF; c*ELMo-BiLSTM*

When training with invoices data with a confidence score more than 90 per receipt document, we continued seeing a visible change in the predictability of NE types of the BiLSTM – CRF and BiLSTM – CNN - CRF model, whereas reasonably stable predictability for the ELMo – BiLSTM model (Figure 29 and 30). The result concurs the robustness of using ELMo – BiLSTM even when there is a change in the underlying data structure. Nonetheless, it is worth noting that in the ELMo – BiLSTM model, the recall for

'merchant' NE type has fallen under 50% (0.49). It indicates that, with high-quality data, we did not obtain any improvement when predicting 'merchant' NEs.

5.1.4 Predict example

This section shows a prediction example using our model against a sample text document extracted from a digital copy of a commercial receipt with the Amazon Textract. The OCR'ed text output is illustrated in Figure 31 below:

```
inputInvoice = 'OUTDOOR, CONCEPTS, Tax Invoice, Outdoor Concepts Ltd, GST
Number 069-712-746, TIME, 11:12PM, Date,  14 Oct 2019, New Zealand,(NZD) Sub
Total, 6539, WEBER PULSE, 1, $349.00, $349.00, 1, $799.00, $799.00, 2000,
7181, Weber Pulse, 1, $69.95, $69.95, $0.00, Premium Cover, 1000/2000, 6415,
Weber Small Drip, 1, $14.95, $14.95, $0.00, Pan, Product Cost:, $1,148.00,
Delivery Details:, Local Oversize $50.00, Sub Total:, $1,198.00, GST:,
$156.26, Tax Invoice Total:, (NZD) $1,198.00, Payments, Method, Ref, Amount,
Total Paid:, (NZD) $1,198.00, 14 Oct 2019, ShopifyV2 - shopify_payments,
$1,198.00, Outstanding:, (NZD) $0.00, Terms: Payment is due before delivery
of goods. For customers on account, invoices are to be paid no later than the
20th of the month following receipt of invoice.'
```

Figure 31: A sample text data for an OCR'ed GST receipt.

After prediction using our trained ELMo-BiLSTM model, unlabelled NE types are identified and labelled with corresponding colours (Figure 32):

```
outdoor , concepts , tax invoice , outdoor concepts ltd , gst number 069 - 712 - 7
46 , time , 11:12pm , date ,   14 oct 2019 , new zealand,(nzd ) sub total , 6539 ,
weber pulse , 1 , $ 349.00 , $ 349.00 , 1 , $ 799.00 , $ 799.00 , 2000 , 7181 , we
ber pulse , 1 , $ 69.95 , $ 69.95 , $ 0.00 , premium cover , 1000/2000 , 6415 , we
ber small drip , 1 , $ 14.95 , $ 14.95 , $ 0.00 , pan , product cost : , $ 1,148.0
0 , delivery details : , local oversize $ 50.00 , sub total : , $ 1,198.00 , gst :
, $ 156.26 , tax invoice total : , ( nzd ) $ 1,198.00 , payments , method , ref ,
amount , total paid : , ( nzd ) $ 1,198.00 , 14 oct 2019 , shopifyv2 - shopify_pay
ments , $ 1,198.00 , outstanding : , ( nzd ) $ 0.00 , terms : payment is due befor
e delivery of goods . for customers on account , invoices are to be paid no later
than the 20th of the month following receipt of invoice .
```

*Figure 32: NER result returned by our ELMo-BiLSTM model. (*Identified NEs are labelled with different colours, Red for Merchant, Blue for GST, Yellow for Time, Green for Date and Purple for Price).*

5.2 Discussion and Limitation

This study seeks to use a deep learning-based method to solve a NER challenge with OCR'ed unstructured NLP data. We began with preprocessing the Amazon Textract JSON output for our commercial receipts to extract only relevant text data; we then created the labelled dataset by annotating text with NE tags with NE positions found in each receipt text data; later we transformed standard NE tags to token-based BIO NE tags; before splitting the data into the train, validation and test dataset, we pad all tokenised receipt data to have the same length as the longest receipt found in our dataset and also to create a vector representation of all word tokens to reduce computational complexity; After we completed the data preprocessing section, we then implemented three different deep learning-based models which are facilitated by the concept of LSTMs, CNNs, CRFs and word embeddings. In the end, model performance and results are evaluated on a range of aspects.

While the results suggest that BiLSTM models with advanced word embedding method are truly capable of identifying NE types from the OCR'ed invoice document and this study is helpful to create an automated and reusable tool for future usage, there are still limitations inherent in the dataset and shortcomings against the overall approach. We highlight five areas for future studies and improvement.

1. This study is aiming to solve a NER task by exclusively depending on the strength of neural networks and statistical models. Rule-based tools such as inference engine and knowledge base are not applied in the process of discovering NE objects. Nevertheless, for entities such as GST number, there is a unique while homogenous linguistic pattern can be easily observed or identified within the data. Well established domain rules and matching patterns may lead to a better result in this scenario. Hybrid solutions that incorporate both the rule-based system and machine learning-based system have shown advantages in several NER studies [36] [37]

[38]. Thus, a combined approach that leveraged by multiple NER methods is worth to explore.

2. The nature of the learning task in this study is fundamentally a supervised learning task. For all supervised machine learning tasks, we hope to establish a feature mapping between the input and output according to examples of input-output pairs from an existing dataset. In this study, we annotate NE in every invoice data entries to create the examples of input-output pairs. However, data labelling, in general, is inevitably labour intensive, resource-consuming and prone to the risk of mislabelling. Therefore, instead of supervised learning, unsupervised or semi-supervised learning techniques can be utilised to mitigate such constraint.

3. This study has emphasised on establishing a feasible process to recognising several key NE types from unstructured commercial invoice data using deep learning models. It did not focus on in-depth analyses on deep learning models themselves. Despite a brief experiment on model reliability was carried out by changing the underlying train/validation/test data structure and the setting of hyperparameters based on [35], we did not evaluate the model performance thoroughly via extensive hyperparameter tuning due to time and resource constraints. Deep learning models possess many hyperparameters. Adjusting settings on these hyperparameters could have a drastic impact on the model performance. As a consequence, the presented model evaluation results in this study may not reflect the true potential of each model.

4. The current NER task is built upon OCR'ed text dataset. Given the rule of 'garbage in, garbage out', the quality of model prediction is highly depending on the quality of input dataset. Nonetheless, OCR processed natural language data inevitability contains noises such as false readings. Since the data is unlabelled, we were not able to obtain the confidence score per each entity types. Therefore, although we evaluated the model performance by comparing using invoice data with high overall OCR confidence score per document versus all available data, we were not

able to eliminate noises per NE types as a high OCR confidence score at the document level does not guarantee a high OCR confidence score at the NE token level. The impact of such noises at the NE level on the model performance was not studied.

5. During the data preprocessing, tokenisation was carried out to turn word phrases into word tokens. However, the tokenisation was done using Python spaCy tokeniser library, which was designed based on structured English language. It tokenises a text document with the assumption that the underlying text data obey the English grammar and syntactic rules. For instance, words are divided into tokens where whitespaces are found, whereas words are not divided into tokens if a dollar sign ($) is found. An immediate problem in this example is that if a 'price amount' entity found in an invoice entry is in '$12' format, it will be treated as one token, whereas if a 'price amount' entity found in an invoice entry is in '$ 12' form, it will be treated as two tokens: '$' and '12'. With '$12' as one token, we lost the contextual relation between '$': a symbol and '12': a digit Perfectly, words should be tokenised with a tailored tokeniser instead of assuming the underlying text follows specific grammar pattern, in this case, it does not.

We hope future work will address these and improve further on the methods presented here.

6

Conclusion

Having an automated system to identify named entities from invoice texts is a critical step in business transaction management and expense analytics. This study participated in a natural language processing pipeline that is aiming to extract named entity information from images of commercial receipts. Despite the fact that antique studies are predominately emphasising on applying deep learning models to identify named entities information from structured text data, we prove that an appropriate deep learning model is also capable of achieving the same goal with unstructured text data that are the output from OCR technologies.

In this study, we adopt a supervised machine learning workflow which is outlined in section 4. We used a variety of Python libraries to annotate OCR'ed receipt text with our chosen named entity labels and to develop three deep learning models to achieve our NER goal: BiLSTM – CRF, BiLSTM – CNN – CRF and ELMo - BiLSTM.

The models are evaluated based on their computational cost, training time, validation accuracy, validation loss and confusion matrices under three different train/validation/test split schemes. We also investigated the impact of OCR noises on the result by training the model using data contain misreadings versus training the model using data with high OCR qualities. We discovered that BiLSTM – CRF is the least expensive one to implement but produces the worst prediction result amongst the three models. The ELMo – BiLSTM with contextualised word embedding consumes the most resources to compute but yields the best prediction result amongst the three models. Meanwhile, BiLSTM – CNN – CRF lies in the middle of the former two models. With the ELMo – BiLSTM model, we retained an above 80% precision, recall and f1-score for most named entities types except for the 'merchant' entity. We also presented a live example to demonstrate how the ELMo – BiLSTM model identifies named entities on an unlabelled receipt text. As a result, we

concluded that even with the presence of OCR noises, our best performing model (ELMo – BiLSTM) is able to produce a reliable NER result and can be reused in a production environment.

In section 5.2, we finally discussed a few limitations encountered during this study. Future recommendations are made based on the discussion around those limitations, include:

1. A hybrid approach that facilitated by a combination of rule-based NER system and deep learning-based NER system may be more suitable to identity certain named entity types.

2. A semi-supervised or unsupervised learning approach may be able to mitigate the hassle of creating a labelled dataset.

3. A more extensive hyperparameter tuning shall be carried out in the future to gain an in-depth understanding of these three deep learning models themselves.

4. For future studies, it is better to examine the influence of OCR noises with more refined details at the word token level rather than using an indicative overall OCR confidence score at the document level.

5. Lastly, we suggest that a purpose-built tokeniser is necessary when performing named entity recognition on unstructured text data that does not follow standard grammar or syntax rules. Future studies shall be aware of the impact of using a non-bespoke tokeniser during the process of creating named entity tokens.

Glossary

MB	Megabyte
GB	Gigabyte
CPU	Central Processing Unit
GPU	Graphic Processing Unit
Token Substitution	Substitute for a word in the context of a clause.
Parsing	Determine the syntactic structure of a text by analysing its constituent words based on an underlying grammar (of the language).
Chucking	Identification of parts of speech and short phrases (like noun phrases).
Recurrent Units	The dimensionality of the output space in a recurrent neural network (Tensorflow).
Recurrent Dropout Rate	Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
Dropout Rate	Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
Epoch	A full iteration over input sample data.
Batch Size	Total number of training examples present in a single batch.
Training Weights	the parameter within a neural network that transforms input data within the network's hidden layers.
Mean residual squared error	an estimator measures the average squared difference between the estimated values and the actual value.
Dense Layer	A densely-connected neural network layer. Dense implements the operation activation(matmul(input, weight) + bias), where weight is a weight matrix, bias is a bias vector, and activation is an element-wise activation function
TimeDistributed Layer	TimeDistributedDense applies a same Dense (fully-connected) operation to every timestep of a 3D tensor.
Relu activation	Rectifier activation function
Kernal	A filter that is used to extract the features from the input data.
Stride	The number of pixels shifts over the input matrix.
Spatial Dropout	A dropout layer drops entire single or multi-dimensional feature maps.
Residual Connection	Allows gradients to flow through a network directly, without passing through non-linear activation functions
Sparse categorical cross-entropy	Computes the cross-entropy loss between the labels and predictions.

Bibliography

[1] P. Dhakal, M. Munikar and B. Dahal, "One-Shot Template Matching for Automatic Document Data Capture," in *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, 2019.

[2] L. Deng and Y. Liu, "A Joint Introduction to Natural Language Processing and to Deep Learning," in *Deep Learning in Natural Language Processing*, L. Deng and Y. Liu, Eds., Singapore, Springer Singapore, 2018, pp. 1-22.

[3] R. M. Weischedel and L. D. Consortium., *OntoNotes release 4.0.*, [Philadelphia, Pa.]: Linguistic Data Consortium, 2011.

[4] B. Black, F. Rinaldi and D. Mowatt, "Facile: Description Of The Ne System Used For Muc-7," 7 2001.

[5] E. Sang and F. Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition," *Proceeding of the Computational Natural Language Learning (CoNLL)*, 7 2003.

[6] Z. Huang, W. Xu and K. Yu, *Bidirectional LSTM-CRF Models for Sequence Tagging*, 2015.

[7] J. P. C. Chiu and E. Nichols, *Named Entity Recognition with Bidirectional LSTM-CNNs*, 2016.

[8] X. Ma and E. Hovy, *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*, 2016.

[9] M. Zhang, G. Geng and J. Chen, "Semi-Supervised Bidirectional Long Short-Term Memory and Conditional Random Fields Model for Named-Entity Recognition Using Embeddings from Language Models Representations," *Entropy*, vol. 22, p. 252, 2 2020.

[10] W. Zaghloul and S. Trimi, "Developing an innovative entity extraction method for unstructured data," *International Journal of Quality Innovation*, vol. 3, p. 3, 2017.

[11] J. Yang, Y. Liu, M. Qian, C. Guan and X. Yuan, "Information Extraction from Electronic Medical Records Using Multitask Recurrent Neural Network with Contextual Word Embedding," *Applied Sciences*, vol. 9, p. 3658, 9 2019.

[12] H. Cho and H. Lee, "Biomedical named entity recognition using deep neural networks with contextual information," *BMC Bioinformatics*, vol. 20, p. 735, 2019.

[13] D. Y. Perwej, S. Hannan, A. Asif and A. Mane, "An Overview and Applications of Optical Character Recognition," *International Journal of Advance Research In Science And Engineering (IJARSE),* vol. Vol. 3, p. Pages 261–274, 6 2014.

[14] M. Cheriet, N. Kharma, C.-L. Liu and C. Y. Suen, "Introduction: Character Recognition, Evolution, and Development," in *Character Recognition Systems: A Guide for Students and Practitioners*, John Wiley & Sons, Ltd, 2007, pp. 1-4.

[15] R. Smith, "An Overview of the Tesseract OCR Engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007.

[16] Amazon, "Amazon Textract Developer Guid," 2020.

[17] L. Chiticariu, Y. Li and F. R. Reiss, "Rule-based information extraction is dead! Long live rule-based information extraction systems!," *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, vol. October, p. 827–832, 1 2013.

[18] K. W. Church and R. L. Mercer, "Introduction to the Special Issue on Computational Linguistics Using Large Corpora," *Computational Linguistics*, vol. 19, p. 1–24, 1993.

[19] T. Young, D. Hazarika, S. Poria and E. Cambria, *Recent Trends in Deep Learning Based Natural Language Processing*, 2018.

[20] P. von Däniken and M. Cieliebak, Transfer Learning and Sentence Level Features for Named Entity Recognition on Tweets, 2017, p. 166–171.

[21] R. Navigli, "Word Sense Disambiguation: A Survey," *ACM Comput. Surv.*, vol. 41, 2 2009.

[22] W. Che and Y. Zhang, "Deep Learning in Lexical Analysis and Parsing," in *Proceedings of the IJCNLP 2017, Tutorial Abstracts*, Taipei, 2017.

[23] L. Ramshaw and M. Marcus, "Text Chunking using Transformation-Based Learning," *ArXiv*, Vols. cmp-lg/9505040, 1995.

[24] J. li, A. Sun, R. Han and C. Li, "A Survey on Deep Learning for Named Entity Recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, p. 1–1, 3 2020.

[25] J. D. Lafferty, A. McCallum and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001.

[26] J. D. Kelleher, "Convolutional and Recurrent Neural Networks," in *Deep Learning - The MIT Press Essential Knowledge series*, MIT Press, 2019, pp. 159-184.

[27] J. Abellan-Abenza, A. Garcia-Garcia, S. Oprea, D. Ivorra-Piqueres and J. Rodríguez, "Classifying Behaviours in Videos with Recurrent Neural Networks," *International Journal of Computer Vision and Image Processing*, vol. 7, p. 1–15, 10 2017.

[28] M. Ali, G. Tan and A. Hussain, "Bidirectional Recurrent Neural Network Approach for Arabic Named Entity Recognition," *Future Internet*, vol. 10, p. 123, 12 2018.

[29] Y. Goldberg and G. Hirst, "Pre-trained Word Representations," in *Neural Network Methods for Natural Language Processing (Synthesis Lectures on Human Language Technologies)*, G. Hirst, Ed., Morgan & Claypool Publishers, 2017, pp. 115-133.

[30] B. Srinivasa-Desikan, "Word2Vec Doc2Vec and Gensim," in *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*, Packt Publishing Ltd., 2018, pp. 199-223.

[31] A. McCallum, D. Freitag and F. C. Pereira, "Maximum Entropy Markov Models for Information Extraction and Segmentation," in *ICML*, 2000.

[32] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, *Deep contextualised word representations*, 2018.

[33] B. Srinivasa-Desikan, "spaCy's Language Models," in *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*, Packt Publishing Ltd., 2018, pp. 33-48.

[34] M. Dwarampudi and N. Reddy, Effects of padding on LSTMs and CNNs, 2019.

[35] N. Reimers and I. Gurevych, "Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks," *ArXiv*, vol. abs/1707.06799, 2017.

[36] X. Li, C. Fu, R. Zhong, D. Zhong, T. He and X. Jiang, "A hybrid deep learning framework for bacterial named entity recognition with domain features," *BMC Bioinformatics*, vol. 20, p. 583, 2019.

[37] K. Shaalan and M. Oudah, "A hybrid Approach to Arabic Named Entity Recognition," *Journal of Information Science*, vol. 40, p. 67–87, 1 2014.

[38] K. S. Bajwa and A. Kaur, "Hybrid Approach for Named Entity Recognition," *International Journal of Computer Applications*, vol. 118, pp. 36-41, 2015.

[39] Y. Goldberg and G. Hirst, "Case Study: A Feed-forward Architecture for Sentence Meaning Inference," in *Neural Network Methods for Natural Language Processing (Synthesis Lectures on Human Language Technologies)*, G. Hirst, Ed., Morgan & Claypool Publishers, 2017, pp. 141-146.

[40] N. Reimers and I. Gurevych, "Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks," 7 2017.