Bi-LSTM-CRF Models for Keyphrase Extraction from Scholarly Documents. Group 12 - IFT 6269 (PGM 2019)

Duy Tân Nguyen [1] **Jun Shao** [2] **Ying Xiao** [3] **François Grégoire-Lacoste** [4]

Abstract

Keyphrase extraction can be addressed as a sequence labeling task. A state-of-the-art approach for this category of tasks is the Bi-LSTM-CRF model which exploits the complementary strengths of conditional random fields (CRF) and bidirectional long-short-term-memory networks (Bi-LSTM). In this project, we intend to reproduce the article of Al-Zaidy et al. (2019) with our customised code to compare the keyphrase extraction performances of a Bi-LSTM-CRF model and of a simple CRF model with the PageRank algorithm on a dataset of scholarly documents. In accordance with the reported results by Al-Zaidy et al. (2019), the Bi-LSTM-CRF model on average outperforms the other two schemes given that it predicts more yet correct keywords. The document-based Bi-LSTM-CRF model works better than the sentence-based Bi-LSTM-CRF model, but there is a big gap between these two models in our implementation, whereas the disparity in Al-Zaidy et al.'s results (2019) is smaller. Our implementation of the CRF and PageRank schemes shows that the former predicts more keyphrases at a lower precision rate, whereas the latter has higher precision score, but overlooks several correct keywords. A major challenge is the limited availability of computational power.

1. Introduction

'*Keyphrase extraction is a key natural language processing task aimed at automatically extracting descriptive phrases or words from a document*' (Hasan & Ng, 2014). In our project, we reproduce the article of Al-Zaidy et al. (2019) that addresses keyphrase extraction as a sequence labeling problem and uses a Bi-LSTM-CRF model to this end. The latter is a neural model that has been commonly used for various other sequence labeling tasks such as Named Entity Recognition. It combines a Bi-LSTM (Long-Short-Term Memory) network to capture the bidirectional long-term dependencies in the text, and a linear-chain conditional random field (CRF) to capture dependencies among the labels.

According to Al-Zaidy et al. (2019), the Bi-LSTM-CRF model performed the best compared to the state-of-the-art methods on various datasets.

We modify the Bi-LSTM-CRF algorithm provided by pytorch (PyTorch-Tutorials) and train the model on the scientific document dataset KP that was provided by Meng et al. (2017).

We also implement the keyphrase extraction model with our customised codes based on PageRank (Wang et al., 2007) and CRF (Zhang et al., 2008) to compare the models of varying sophistication. Al-Zaidy et al. (2019) also compared their Bi-LSTM-CRF model with these algorithms in their paper.

2. Methodology

2.1. Conditional Random Fields: Generalities

Conditional Random Fields are comprised of a Markov random field over a set V_y of hidden variables y_i that is conditioned over a set of observations x_i (Sutton & McCallum, 2012). The conditional probability distribution of the hidden state sequence $\mathbf{y}$ given the observations $\mathbf{x}$ has the form:

$$p(\mathbf{y}|\mathbf{x};\mathbf{W},b) = \frac{1}{Z(\mathbf{X})}\exp\left(\sum_{y\in V_y}\mathbf{W}^{\top}_{y_{i-1},y_i}\mathbf{X}_i + \mathbf{b}_{y_{i-1},y_i}\right)$$

The CRF can in general be represented either as a partially directed graphical model or as an undirected graphical model (UGM) (Figure 1). In the former case, the set of Y itself forms a UGM while the edges that represent the condi-

*Equal contribution [1]Student ID: 20168919 [2]Student ID: p1239682 [3]student ID: p1183562 [4]Student ID: p0639518. Correspondence to: Duy Tân Nguyen <duy-tan.nguyen@hec.ca>, Jun Shao <jun.shao2@mail.mcgill.ca>, Ying Xiao <xiaoyingtutu@gmail.com>, François Grégoire-Lacoste <fglacoste@gmail.com>.

IFT 6269 project report - Université de Montréal

tionals of Y given X are shown as directed edges ($X \rightarrow Y$) (Figure 1).

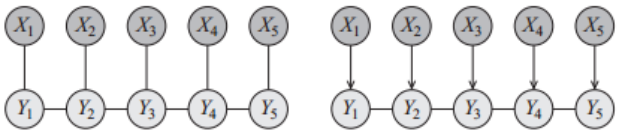

Figure 1. Representation of CRF as UGM and partially DGM (figure taken from Koller & Friedman, 2009, p. 143)

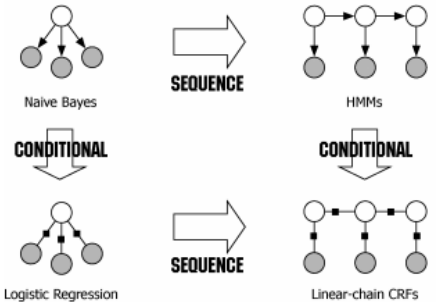

Figure 2. Analogy between naive Bayes, logistic regression, HMM, and CRF (figure taken from Sutton & McCallum, 2012, p. 86)

The representation of the CRF as a UGM allows an analogy (shown in Figure 2) between undirected discriminative models (CRF) and directed generative models (Naïve Bayes, HMM). The linear-chained CRF and HMM models are then contrasted based on these properties of their graphical models, while being similar by both representing sequences of hidden states in the presence of certain observations. A major advantage of the linear-chained CRF over the HMM is the relaxation of assumptions on the conditional independence between the observations, and thus, the CRF model can handle overlapping features and datasets of unknown joint distribution. The multinomial logistic regression can thus be regarded as '*the simplest kind of CRF*' (Sutton & McCallum, 2012).

Training is performed via Maximum Likelihood Estimation with Stochastic Gradient Descent (SGD). More precisely, we estimate model parameters $\mathbf{W}$ and $\mathbf{b}$ from a training dataset D by maximizing the log-likelihood:

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{j=1}^{n} \log p\left(y^{(j)} \mid x^{(j)}; \mathbf{W}, \mathbf{b}\right)$$

Label prediction (also called decoding) is performed via the Viterbi algorithm like that of an HMM. It allows the CRF layer to jointly decode the best sequence of labels given the input sequence instead of decoding each label independently.

2.2. Long Short Term Memory networks

The keyphrase extraction problem can be devised as a sequence labelling task, meaning that it involves the understanding of prior words. Traditional NNs cannot handle this relationship, but Recurrent Neural Networks (RNNs) can. Their loops allow passing information from one step to another. LSTM is a special kind of RNNs. It is specifically formulated to handle the RNN issue of long-term dependence (Hochreiter & Schmidhuber, 1997), and outperforms the standard version in practice (Sagheer & Kotb, 2019).

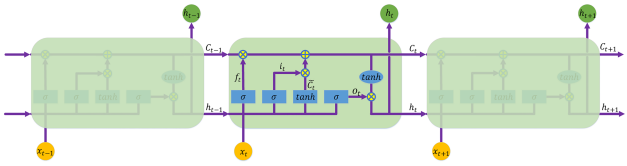

Figure 3. The repeating module in an LSTM contains four interacting layers (figure taken from Bao et al., 2017)

LSTM also has a chain of repeating modules like all other RNNs, but the structure of these modules are different (shown in Figure 3) in that there are four interacting neural network layers instead of one.

The integral part of LSTMs is the cell state, which resembles a conveyor belt with the top line transcending the diagram in Figure 3. Information can be erased or added to the LSTM cell via '*gates*.' Gates are made up of a sigmoid layer and an elementwise multiplication operator. This layer output is between zero and one, indicating the degree to which information is let through its gate. There are three types of gates in a typical LSTM model, namely '*forget gate*,' '*input gate*' and '*update gate*' for cell state protection and control (Fischer & Krauss, 2018).

First, the LSTM model decides which information to be '*forgotten*' by the cell state. This decision is obviously taken by the '*forget gate layer*.' Based on the hidden state (h_{t-1}) from the preceding cell state (C_{t-1}) and the input (x_t) of the current cell state, the function outputs f_t which is the proportion of information from the previous cell state to be kept at this cell state.

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

In the next step, the model decides which new values are retained in the cell. The first sub-step involves the '*input gate layer*' deciding whether the new information is inputted into the cell ($i_t = 1$) or not ($i_t = 0$). Next, a hyperbolic tangent function of the update gate composes a vector of candidate values, $\tilde{C}_t$, for the memory cell.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The third step combines the previous two steps to update the new cell state (C_t) from the old one (C_{t-1}). The elementwise product of f_t and C_{t-1} denotes how much to keep from the previous state, and the one for i_t and $\tilde{C}_t$ shows which new information to load into the cell state. The sum of these two indicates the decided updates.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Finally, the output (o_t) is a version of the cell state (C_t) filtered by a sigmoid function. Then, the cell state is rescaled from -1 to 1 through a hyperbolic tangent function and an elementwise multiplication is applied to yield the new hidden state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

Bidirectional LSTMs are an extension of traditional LSTMs. It trains two instead of one LSTM on the input sequence (Figure 4). The first recurrent layer processes the input sequence as-is (forward) and the second layer duplicates the first layer in a backward manner. The two layers in parallel give further contextual information to the model for performance improvement, especially in sequence classification.

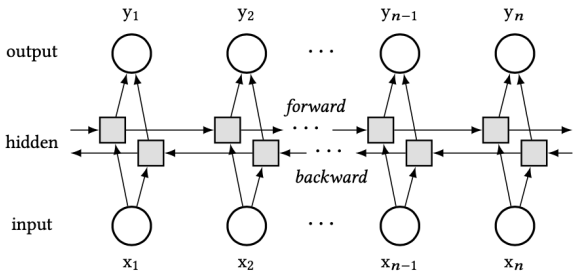

Figure 4. Layers in a Bi-LSTM network (figure taken from Al-Zaidy et al., 2019, p. 2553)

2.3. The Bi-LSTM-CRF model

As shown in Figure 5, there are two primary layers in the Bi-LSTM-CRF model. The first layer is a Bi-LSTM network, which aims at capturing the semantics of the input text sequence. After a fully-connected layer, the output of the Bi-LSTM layer is passed to the CRF layer, which produces a probability distribution over the tag set for each word of the entire sequence.

Partition functions are calculated by the forward algorithm (α-recursion) in log space using predictive labels (soft assignment) and ground-truth labels, respectively. The loss function is the gap between these two partition functions. The Viterbi algorithm is deployed to generate the label sequence with maximum likelihood.

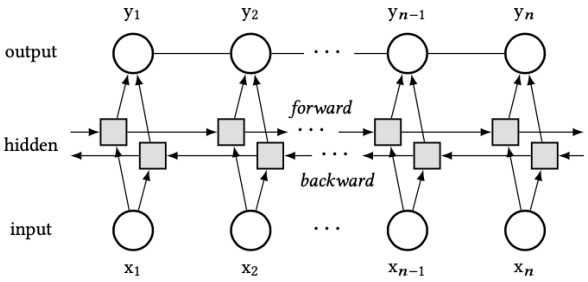

Figure 5. Layers in a Bi-LSTM-CRF network (figure taken from Al-Zaidy et al., 2019)

3. Dataset Preparation

In their article, Al-Zaidy et al. (2019) use three datasets: KP, WWW and KDD. We only use the KP dataset that was made available by Meng et al. (2017) and was collected from several online libraries, namely Wiley, ACM and Web of Science. Each document of the KP dataset contains titles, abstracts, and author-assigned keyphrases. The KP dataset was split into three parts: 527k training dataset, 20k validation and 20k test dataset. We use the whole dataset for Bi-LSTM-CRF model and a sample dataset for the CRF and PageRank model.

New functions in python have been created for cleaning, tokenizing and vectorizing the data. The data are in the format of text files that place abstracts, titles and keyphrases in separate subsections. These text data need to be cleaned of brackets and descriptive words such as 'abstract', 'id' and 'tgt.' We wrote some of the simple data-preprocessing functions to ensure that the predicted keyphrases can be retrieved from the coded data.

3.1. The CRF model

For our own CRF model, we use the small version for the English language of the 'spacy' package to divide the document text into a sequence of single words (tokens), whose part of speech can be identified so that we can extract their features and retain the words most probable to be keywords. This package also helps us identify noun phrases in the document text so that we can determine which words can be combined into a phrase when decoding.

Training with all of these datasets is computationally expensive for the CRF experiment and thus we focus on a sample of these datasets. The sample dataset is split as follows: 100 in validation set, 100 in testing set and 1000 in training set.

3.2. The Bi-LSTM-CRF model

In our work, the Bi-LSTM-CRF model is trained on the whole training dataset, with validation set for model selec-

tion. The final model is evaluated on all three datasets.

After stop words filtering and low frequency words filtering (frequency<3), words collected from the training and validation datasets are used to build a vocabulary with size of 148,682. Two dictionaries are created to link each token in the vocabulary with their ID. Words that are not included in the vocabulary are represented by an 'unknown' symbol. Two special symbols are used to represent the start and end of each inputted document or sentence.

After that, all documents of three datasets are converted into their corresponding index sequence and label sequence pairs. Each index-label sequence pair is padded to same size so that they can be processed in batch. To reduce the space for padding, we used four thresholds (lower average, average, higher average and maximum) to standardize the length of documents during data preprocessing. Thus, we generated four data-loaders, each data-loader contains padded documents with same length.

Besides, we do the same transformation for each sentence so that we can evaluate the sentence-level Bi-LSTM-CRF model as well.

4. Implementation

4.1. Linear-Chain CRF for Keyphrase Extraction

4.1.1. Feature preparation

To compute the CRF model alone, we follow the paper of Zhang et al. (2008) and create a sequence of words, where we retain only nouns, adjectives, verbs and proper names. Then, we create feature matrix $\mathbf{X}_{T\times|F|}$, where
T denotes the number of words in the sequence;
$|F|$ denotes the number of features;
$\mathbf{X}_{\cdot 1}$ denotes whether there is a word before the word being considered;
$\mathbf{X}_{\cdot 2}$ denotes whether there is a word after the word being considered;
$\mathbf{X}_{\cdot 3}$ denotes whether the word being considered is a noun;
$\mathbf{X}_{\cdot 4}$ denotes whether the word being considered is an adjective;
$\mathbf{X}_{\cdot 5}$ denotes whether the word being considered is a verb;
$\mathbf{X}_{\cdot 6}$ denotes whether the word being considered is a proper name;
$\mathbf{X}_{\cdot 7}$ denotes whether the word being considered is in a noun phrase in the document;
$\mathbf{X}_{\cdot 8}$ denotes whether the word being considered is in the document title;
$\mathbf{X}_{\cdot 9} = \frac{count_i^d}{\sum_j count_j^d} \cdot \ln \frac{|D|}{\sum_{d'}^{D} \mathbb{I}\{count_i^{d'}>0\}}$ denotes the TFIDF (Salton & Buckley, 1988) of the word being considered, where $count_i^d$ denotes how many times word i appears in document d and D is the set of all documents in the training set;
$\mathbf{X}_{\cdot 10} = \frac{pword_i^{(1)}}{T}$ with $pword_i^{(1)}$ denoting the position of the first appearance of the word being considered in the document;
$\mathbf{X}_{\cdot 11} = \frac{pword_i}{pword_i^{(1)}}$ denotes whether the same word has appeared before in the document;
$\mathbf{X}_{\cdot 12} = 1$ denotes the constant term or bias.
The hidden state in our case is whether the word is included in the keyphrase or not ($S = \{0, 1\}$). In comparison the poster version, the model reported here utilises more features and has categorical variables converted into sets of binary variables.

4.1.2. Model training

Zhang et al. (2008) used Maximum Entropy learning algorithm to train their CRF model. Given the concavity of the model, we apply SGD to train our CRF model as per the guidance of Sutton & McCallum (2012). Indeed, Sutton & McCallum (2012) and Zhang et al. (2008) use slightly different notation from our selected paper (Al-Zaidy et al., 2019). For consistency, we adapt the notation of Al-Zaidy et al. (2019) to demonstrate the log likelihood as follows

$$\mathcal{L}(\mathbf{W}) = \sum_{d=1}^{|D|}\sum_{t=1}^{T} \mathbf{W}_{y_{t-1},y_t}^{\top}\mathbf{X}_t^d - \sum_{d=1}^{|D|} \ln Z\left(\mathbf{X}^d\right) - \sum_{f=1}^{|F|}\frac{w_f^2}{2\sigma^2}$$

with w_f being the fth element of vector $\mathbf{W}_{y_{t-1},y_t}$ and $\frac{1}{2\sigma^2}$ being the regularisation parameter where Sutton & McCallum (2012) often use $\sigma^2 = 10$.

To compute the normalisation constant $\ln Z\left(\mathbf{X}^d\right)$, we follow the forward-backward algorithm in Sutton & McCallum (2012) below

$$\alpha_t(j) = \sum_{i\in S} \exp\left(\mathbf{W}_{i,j}^{\top}\mathbf{X}_t\right)\alpha_{t-1}(i)$$
$$\beta_t(i) = \sum_{j\in S} \exp\left(\mathbf{W}_{i,j}^{\top}\mathbf{X}_{t+1}\right)\beta_{t+1}(j)$$
$$Z(\mathbf{X}) = \beta_0(y_0) \text{ and } Z(\mathbf{X}) = \sum_{i\in S}\alpha_T(i)$$

with $\alpha_1(j) = \exp\left(\mathbf{W}_{0,j}^{\top}\mathbf{X}_1\right)$ and $\beta_T(i) = 1$. The log operator is used to avoid underflows during implementation.

With SGD, the idea is to randomly choose a document in the training dataset and update the weights in the direction of that instance's gradient (Sutton & McCallum, 2012). In particular, the mth update of the weights is computed as

follows

$$\mathbf{W}^{(m)} = \mathbf{W}^{(m-1)} + \frac{1}{\sigma^2(m_0+m)}\nabla\mathcal{L}_d\left(\mathbf{W}^{(m-1)}\right)$$

$$\frac{\partial\mathcal{L}_d}{\partial w_f} = \sum_{t=1}^{T}\mathbf{X}_{t,f}^{(d)} - \sum_{t=1}^{T}\sum_{i,j\in S}\mathbf{X}_{t,f}^{(d)}p\left(i,j \mid \mathbf{X}^{(d)}\right) - \frac{w_f}{|D|\sigma^2}$$

$$p\left(y_{t-1}^{(d)}, y_t^{(d)} \mid \mathbf{X}^{(d)}\right) = \frac{1}{Z\left(\mathbf{X}^{(d)}\right)}\alpha_{t-1}\left(y_{t-1}^{(d)}\right) \cdot \exp\left(\mathbf{W}_{y_{t-1},y_t}^{\top}\mathbf{X}_t^{(d)}\right)\beta_t\left(y_t^{(d)}\right)$$

with m_0 being fine-tuned based on one pass over a random subset of the training documents with a set of randomly generated step sizes, amongst which the one resulting in the highest likelihood is chosen.

Given that the gradient of some documents can point to a direction very different from the optimum, we decide to update the weight parameters based on approximately every 10 documents in the training set if the batch average F1 score is less than 0.5 or the loss value is greater than 0.001. The training dataset is divided into batches of 100 documents. Once a batch training is completed, the model is tested on the validation and testing dataset. If the average F1 score of the validation set is less than 0.5, the weight parameters are then re-updated based on 20 documents in the training batch before. The test on the testing set is conducted afterwards for performance tracking only. No update is triggered by the performance on the testing set.

Finally, the Viterbi algorithm is applied to decode the sequence and select the words likely to be included in the author keyphrases.

As the SGD method can get stuck in local optima, we re-run the model 20 times (20 episodes), each with different initialisation for 100 epochs (100 scans over the whole training dataset) and report the average performance along with its variance. The model is run on the Graham cluster of Compute Canada.

Given the computational intensiveness of the CRF model, we only ran this model on a thousand training, a hundred validation and a hundred testing documents, which is a subset of the original dataset of Meng et al. (2017).

4.2. PageRank

As a baseline, we use the PageRank method to compute the conditional probability that a word in the text is included in the keywords given that its preceding word is already a keyword. We create the transition matrix $\mathbf{A}_{W\times W}$ where $\mathbf{A}_{ij} = p\left(j \mid i\right)$ denotes the probability that word j follows word i within a window of three words. $\mathbf{A}_{ii} = 0, \forall i \in \{1,\ldots,W\}$ because a word cannot follow itself in a phrase. W is the set of unique nouns, adjectives, verbs and proper names in the text. Each entry of $\mathbf{A}$ is computed as per the sigmoid function used by Zhang et al. (2008). Then, $\mathbf{A}$ is normalised so that the sum of each row equals 1 and a damping factor $d = 0.15$ is added to each entry of $\mathbf{A}$ as follows

$$\mathbf{A}_{ij} = (1-d)\,\mathbf{A}_{ij} + d\frac{\mathbb{I}\left\{i,j \in \text{phrases}\right\} + \mathbb{I}\left\{i,j \in \text{title}\right\}}{\sum_{j'}\mathbb{I}\left\{i,j' \in \text{phrases}\right\} + \mathbb{I}\left\{i,j' \in \text{title}\right\}},$$

where $\mathbf{A}_{ij} = \frac{1}{1+\exp\left(-\mathbf{w}^{\top}\mathbf{X}_{i,:,j}\right)}, \forall i \neq j$ and $\mathbf{X}_{W\times|F|\times W}$ is a tensor of features $\in F$ related to word i and word j, including how many nouns, adjectives, verbs or proper names before and after word i, the part of speech of word i and word j, how many times word i is followed by word j in the document text and whether they appear in the title or in the same noun phrase we compile from the document text.

Next, we initialise vector $\mathbf{v}_W$ of marginal probability that word i is included in the keywords. We compute $\mathbf{v} = \mathbf{v}^{\top}\mathbf{A}$ until convergence.

Finally, the Viterbi algorithm is used to decode phrases of up to three words that are the most probable to be included in the keywords. With the decoded keyphrases, we use Newton's update to train $\mathbf{w}$. The dataset for the PageRank model is the same as that of our CRF scheme. We also run this model on the Graham cluster of Compute Canada.

4.3. Bi-LSTM-CRF Keyphrase Extraction

The Bi-LSTM-CRF model takes the keyphrases extraction problem as a sequence labeling task. The abstract-keyphrases data pairs are converted into a sequence of word tokens paired with a label sequence of equal length. Each word token has a KP label if it occurs in a keyphrase, otherwise NKP. We also set start, padding and end symbols in an entire sequence.

Each document is passed to the network as input after being converted into a word embedding with 100 dimensions. The word embedding is initialized with a Glove pre-trained word embedding matrix (Pennington et al., 2014).

We train the Bi-LSTM-CRF model on both document-level and sentence-level inputs so that we can evaluate their performance difference.

We use Stochastic Gradient Descent with the initial learning rate $Lr = 0.015$ and decay ratio Lr-decay$= 1e{-}4$. The best F1-score on validation dataset is used for model selection. Both the document-based model and sentence-based model are trained on a Tesla V100 GPU for 75h (3 epochs).

Our implementation of the network model is based on a modified version of the code developed by Pytorch (PyTorch-Tutorials).

5. Results

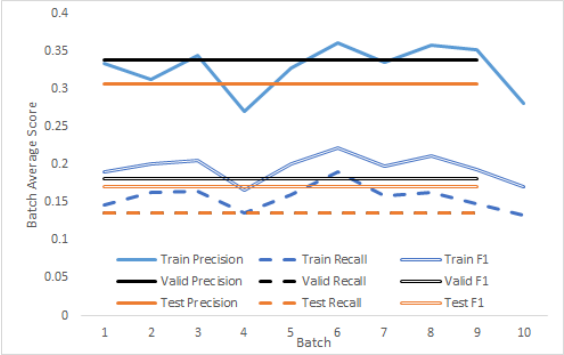

Figure 6. PageRank Batch Average Score

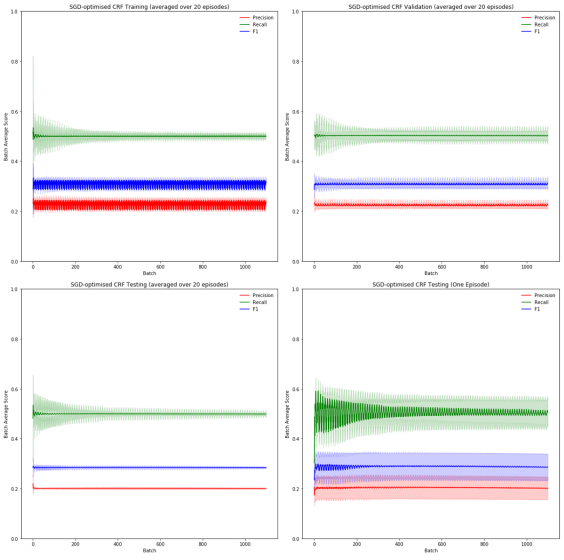

Figure 7. CRF Batch Average Score

5.1. PageRank

The PageRank approach is simple, but its Precision is still higher than that of our CRF scheme, which was averaged in the last epoch over 20 episodes. Our PageRank Precision scores in the validation and test sets are higher than the 0.153 figure reported by Al-Zaidy et al. (2019), but its Recall rate is lower. Overall, the F1 score of our PageRank scheme stands at 0.182 for the validation set and 0.172 for the test set, which are not far from their result of 0.184. However, our PageRank scheme did not seem to learn from the training set as depicted in Figure 6. Still, there is overall no palpable overfitting in the result.

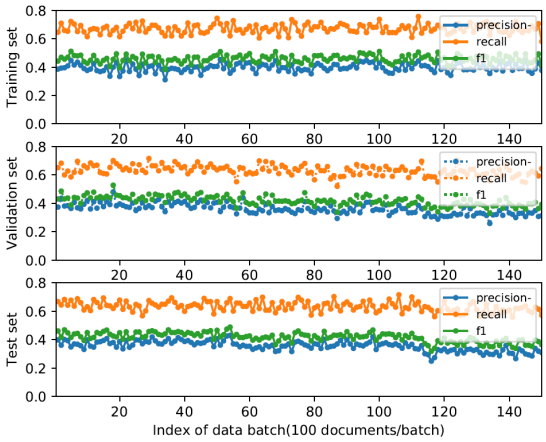

Figure 8. Batch Average Score of Document-based Bi-LSTM-CRF

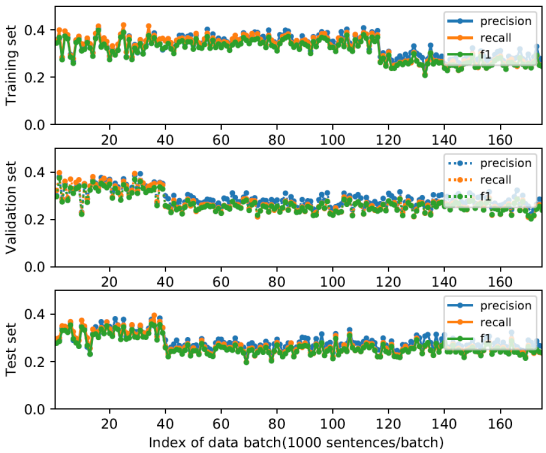

Figure 9. Batch Average Score of Sentence-based Bi-LSTM-CRF

5.2. CRF

With regard to our CRF model, it predicted several keywords, but the Precision score is lower than those of the other two methods (Table 1). As can be seen in Figure 7, the variation across episodes owing to different initialisation was not large, but the improvement over iterations was also marginal. It is probable that we did not choose a proper step size even though this parameter was fine-tuned based on a randomly selected subset of the training data at the beginning of each episode. Vis-à-vis the results of Al-Zaidy et al. (2019), our CRF Precision is lower than their 0.667 Precision, but with more keyphrases predicted, our Recall rate is higher and thus the F1 score of our linear-chain CRF model is higher, standing at 0.308 for the validation dataset and 0.284 for the testing set, compared to their 0.175 result. On a side note, the gaps between training, validation and testing results are small, implying that the model did not *overfit*.

Table 1. Result Summary

Score	bi-LSTM-CRF sent-	bi-LSTM-CRF doc-	CRF	PageRank
Train Precision	0.332	0.398	0.227	0.328
Train Recall	0.323	0.670	0.500	0.156
Train F1	0.311	0.460	0.309	0.196
Valid. Precision	0.290	0.359	0.224	0.339
Valid. Recall	0.272	0.631	0.502	0.136
Valid. F1	0.265	0.418	0.308	0.182
Test Precision	0.285	0.355	0.201	0.307
Test Recall	0.270	0.633	0.499	0.136
Test F1	0.262	0.416	0.284	0.172

Note:
*'sent-' *stands for sentence-based Bi-LSTM-CRF model*
*'doc-' *stands for document-based Bi-LSTM-CRF model*
**Due to the constraint of computational resources, the document-based model was tested on training dataset for only 1247 iterations (100 documents/iteration), while the sentence-based model was tested on training dataset for 1593 iterations (1000 sentences/iteration).*

5.3. Bi-LSTM-CRF

First, we evaluated the performance of document-based model on part of training dataset, validation dataset and test dataset. As shown in Table 1 and Figure 8, the document-based model achieved similar results on validation dataset and test dataset for Precision, Recall and F1 scores. The batch average Precision, Recall and F1 scores on test dataset are 0.355, 0.633 and 0.416, respectively. The performance on the training dataset is slightly better than those on validation and test dataset, with an average Precision of 0.398, Recall 0.670 and F1-score 0.460. The narrow gap between the scores on training set and those on validation/test dataset shows that the model is not overfitting. After all, we only trained the document-based Bi-LSTM-CRF for 3 epochs.

Compared with the result of Al-Zaidy et al. (2019) that is Precision 0.673, Recall 0.302 and F1-score 0.418, our experiment achieved a higher recall but lower precision. That means our document-based model tended to cover as many keyphrases as possible. However, the F1-score is very similar between these two experiments.

The results of sentence-based Bi-LSTM-CRF model are shown in Table 1 and Figure 9. The model achieved a very similar result on validation dataset and test dataset. The batch average Precision, Recall and F1-score on test dataset are 0.285, 0.270 and 0.262, respectively. The performance on the training dataset is higher than those on validation/test dataset, obtaining a batch average Precision of 0.332, Recall 0.323 and F1-score 0.311. As shown in Figure 9, there was an obvious drop of performance on the validation and test datasets since iteration 38. The drop point also occurred on training dataset. It originates from the change in sentence length for evaluation. To reduce the space for padding, we used four thresholds (lower average=2, average=6 , higher average=10 and maximum=65) to standardize the length of sentences during data preprocessing. Thus, we generated four data-loaders, each contains padded sentences with same length. The first data-loader contains sentences with a maximum length of 2 (excluding start and end symbols in the sequence), thus easier to be labelled correctly.

Compared with the result of Al-Zaidy et al. (2019) that is Precision 0.64.2, Recall 0.2472 and F1-score 0.356, our experiment achieved lower results. This may have been caused by the quality of sentence generation of the '*spacy*' tool (spacywebsite).

Finally, given the results of two models in our experiments, the document-based model outperforms its sentence-based counterpart, achieving an improvement over 15% on F1-score. This is likely due to the increased information captured by the bi-LSTM layer when using the entire document instead of each sentence.

As shown in Table 1, the document-based Bi-LSTM-CRF outperforms the CRF and PageRank model, with a gap of 0.132 and 0.244, respectively on F1-score.

6. Possible Improvement via Sampling

To better approximate the probabilistic distribution, we tried to utilise Gibbs Sampling in accordance with the paper of Liu et al. (2012), which was claimed to be able to predict keywords that were not mentioned in the document text. This is called '*keyphrase identification*,' which includes '*keyphrase extraction*.' In fact, some words or phrases in the document text may trigger or refer to certain words or phrases that can, albeit not mentioned in the text, describe the overarching topic of the document, hence, likely to be keywords/keyphrases (Liu et al., 2012). Yet, given the large corpus of the training dataset, the sampling is computationally heavy, and we have not got any results so far. A solution would be to rely on a specialized library such as pyMC3 or pyro for their efficient implementations of sampling-based inference. A Bayesian approach of the CRF with these tools would also allow assessing the uncertainties and further address overfitting (Sutton & Zhang, 2010).

7. Conclusion

Our project implemented the Bi-LTSM-CRF model of Al-Zaidy et al. (2019) along with PageRank and CRF schemes, which were amongst the algorithms Al-Zaidy et al. (2019) used for performance comparison. We used some data-preprocessing packages for text vectorisation and tokeni-

sation. Although the time and computational limits did not allow us to run the models for too many epochs over the full dataset and there are some discrepancies between their reported performance and our results, the overall trend is in accord with their claim that the document-based Bi-LSTM-CRF outperforms the other algorithms. Indeed, our F1 scores for document-based Bi-LSTM-CRF and PageRank are similar to their counterparts in Al-Zaidy et al.'s results (2019). The F1 score of our CRF model is higher than that of Al-Zaidy et al. (2019), but its Precision is much lower. Our implementation of the sentence-based Bi-LSTM-CRF could not reach the level reported by Al-Zaidy et al. (2019) and there is a big disparity between the two Bi-LSTM-CRF models we implemented. Possible reasons include the probable inadequacy of contextual information provided by a single sentence for effective keyphrase extraction, the quality of sentence separation for data-inputting, and insufficient training for both models.

Besides, both the training of document-based model and sentence-based model are time consuming, taking an average of 25h for 1 epoch. Modification of the code for parallel computing in all functions is likely to reduce the training time.

Finally, there are some potential measures to improve the current performance of our model, including Gibbs Sampling and Bayesian Inference, of which we hope to complete the ongoing implementation to further develop this project.

8. Acknowledgment

Our special thanks to Prof Simon Lacoste-Julien and Mr José Gallego (TA) for their guidance during the course.

The hyperlinks to the codes yielding the results reported herein can be found below

Bi-LSTM-CRF

Linear-chain CRF - 3 versions

PageRank Algorithm

CRF-PageRank small data

Full Dataset

References

Al-Zaidy, R. A., Caragea, C., and Giles, C. L. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *Proceedings of The Web Conference (WWW 2019)*, pp. 1207–1216, 2019.

Bao, W., Yue, J., and Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):1–24, 2017.

Fischer, T. and Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

Hasan, K. S. and Ng, V. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1262–1273, 2014.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, Massachusetts, USA, 2009.

Liu, Z., Liang, C., and Sun, M. Topical word trigger model for keyphrase extraction. In *Proceedings of COLING 2012*, pp. 1715–1730, 2012.

Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., and Chi, Y. Deep keyphrase generation. In *Proceedings of the ACL*, pp. 582–592, 2017.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

PyTorch-Tutorials. Advanced: Making dynamic decisions and the bi-lstm crf. URL https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html?highlight=crf. [Accessed 5th Dec 2012].

Sagheer, A. and Kotb, M. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523, 1988.

spacywebsite. Industrial-strength natural language processing. URL https://spacy.io/.

Sutton, C. and McCallum, A. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012.

Sutton, C. and Zhang, Y. Adaptive mcmc methods for bayesian conditional random fields, 2010. URL https://montecarlo.wikidot.com/local--files/contributed-abstracts/nipsmc2010_sutton_zhang.pdf.

Wang, J., Liu, J., and Wang, C. Keyword extraction based on pagerank. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 857–864, 2007.

Zhang, C., Wang, H., Liu, Y., Wu, D., Liao, Y., and Wang, B. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4(3):1169–1180, 2008.