Bi-LSTM-CRF Sequence Labeling for Keyphrase Extraction from Scholarly Documents

Rabah A. Al-Zaidy
Pennsylvania State University
alzaidy@psu.edu

Cornelia Caragea
University of Illinois at Chicago
cornelia@uic.edu

C. Lee Giles
Pennsylvania State University
giles@ist.psu.edu

ABSTRACT

In this paper, we address the keyphrase extraction problem as sequence labeling and propose a model that jointly exploits the complementary strengths of Conditional Random Fields that capture label dependencies through a transition parameter matrix consisting of the transition probabilities from one label to the neighboring label, and Bidirectional Long Short Term Memory networks that capture hidden semantics in text through the long distance dependencies. Our results on three datasets of scholarly documents show that the proposed model substantially outperforms strong baselines and previous approaches for keyphrase extraction.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Information extraction**.

KEYWORDS

Keyphrase extraction, sequence labeling, deep learning.

ACM Reference Format:
Rabah A. Al-Zaidy, Cornelia Caragea, and C. Lee Giles. 2019. Bi-LSTM-CRF Sequence Labeling for Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313642

1 INTRODUCTION

Keyphrase extraction is a key natural language processing task aimed at automatically extracting descriptive phrases or words from a document [17]. Keyphrases (also referred as keywords) provide a brief summary of the content of a document. The importance of keyphrases has been widely recognized in many downstream applications such as query formulation, document clustering, classification, recommendation, indexing, and summarization [16, 21, 38].

Most of the existing works on automatic keyphrase extraction focus on supervised and unsupervised approaches. The unsupervised approaches use ranking techniques to rank phrases based on the aggregated "informativeness" scores of the individual words comprising a phrase. Graph-based ranking algorithms that are applied to the word graph representation of a document are the most prevalent in this category. To construct the graph, each candidate word in a document (i.e., a word with certain part-of-speech tags) is mapped to a node and connecting edges represent the association patterns among the candidate words. The scores of individual words are estimated using various graph centrality measures such as PageRank [10, 12, 27, 34, 41].

The supervised approaches use binary classification to label candidate phrases positively (as keyphrases) or negatively (as non-keyphrases), based on a set of linguistic and statistical features such as *tf-idf*, part of speech (POS) tags, and the position of phrases in documents. Supervised keyphrase extraction allows for expressive feature design and is reported to often outperform unsupervised methods [6, 23]. Two major limitations of these supervised approaches are: (1) they classify the labels of each candidate phrase independently, while completely ignoring the dependencies that could potentially exist between neighboring labels; and (2) they do not incorporate the hidden semantics in the input text.

More recently, Gollapalli et al. [13] formulated keyphrase extraction as sequence labeling and showed that using linear-chain Conditional Random Fields can improve the performance over baseline models for this task. However, the approach in [13] does not explicitly take into account the long-term dependencies and semantic relationships hidden in text. Figure 1 shows examples of long-term dependency patterns and semantic relationships hidden in text from a research paper published in the ACM KDD conference, e.g., the phrase "we describe" is followed by (and indicative of) keyphrases, whereas the terms "Conditional Random Field" and "discriminatively-trained model" are semantically related. We posit that a deep understanding of the text is required in order to correctly identify keyphrases for a document.

To this end, we address keyphrase extraction as a sequence labeling problem, using *research papers* as a case study, and precisely aim to capture both the semantics of document contexts as well as the dependencies among the labels of neighboring words in order to overcome the limitations in previous approaches. Specifically, we explore a neural learning model, called Bi-LSTM-CRF, that combines a bi-directional Long Short-Term Memory (Bi-LSTM) layer to model the sequential text data with a Conditional Random Field (CRF) layer to model dependencies in the output [19, 30]. The result of this extraction task will aid indexing of documents in scholarly document collections, and hence, will lead to improved organization, search, retrieval, and recommendation of scientific documents. In summary, our contributions are as follows:

- We explore a neural learning model for keyphrase extraction from scholarly documents that combines the complementary strengths of Bi-LSTM and CRF. In the combined model, the input and output layers are not directly connected as in CRF, but instead a Bi-LSTM layer is inserted between them to exploit the long term dependencies in the text.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313642

A Unified Approach for **Schema Matching**, **Coreference** and **Canonicalization**
by Michael L. Wick, Khashayar Rohanimanesh, Karl Schultz and Andrew McCallum

The automatic consolidation of database records from many heterogeneous sources into a single repository requires solving several *information integration* tasks. Although tasks such as **coreference**, **schema matching**, and **canonicalization** are closely related, they are most commonly studied in isolation. Systems that do tackle *multiple integration problems* traditionally solve each independently, allowing errors to propagate from one task to another. *In this paper, we describe* a *discriminatively-trained model* that reasons about **schema matching**, **coreference**, and **canonicalization** jointly. [...]

Author-annotated keyphrases: *Data Integration, Coreference, Schema Matching, Canonicalization, Conditional Random Field, Weighted Logic*

Figure 1: The title and abstract of a paper by Wick et al. (2008) and the author-input keyphrases for the paper. Cyan bold phrases represent the gold-standard (author-annotated) keyphrases for the document. Red italic phrases represent long-term dependency patterns or semantic relationships with the gold-standard keyphrases.

- We conduct a thorough evaluation to examine the role of each layer in the model. To our knowledge, there is no study that demonstrates an ablation experiment that compares, with this clarity, the role of input dependencies and label dependencies in keyphrase extraction from research papers.
- We show empirically on three datasets of research papers that the Bi-LSTM-CRF model outperforms strong baselines and previous works on the keyphrase extraction task.
- We investigate the performance of Bi-LSTM-CRF at document and sentence level and show that a document level model that captures a broader context is more accurate than a sentence level model. To our knowledge, this has not been addressed before, especially, not on a large scale dataset as we do in this paper.

In the next section, we describe related work on keyphrase extraction. We then introduce the neural learning model in Section 3, followed by Section 4 that presents our evaluation setup. Finally, we discuss our experimental results in Section 5 before we conclude the paper and touch on future directions in Section 6.

2 RELATED WORK

Keyphrase extraction has been the focus of many studies. These studies generally adopt a two phase approach. In the first phase, candidate words or phrases are extracted from the text using heuristics such as POS patterns for words or *n*-grams [20]. In the second phase, the candidate phrases are predicted as keyphrases or non-keyphrases, using both supervised and unsupervised approaches. In the supervised approaches, the prediction is done based on a selection of features, e.g., POS tags, *tf-idf* scores, and position information, used in conjunction with machine learning classifiers [11, 20, 36, 39]. Traditional features were also combined with features extracted from external sources such as WordNet and Wikipedia [29, 32] or from various neighborhoods, e.g., a document's citation network or a webpage's hyper-link network [5, 6, 22].

Unsupervised approaches include phrase scoring methods based on measures such as *tf-idf* and topic proportions [3, 28, 46], graph-based ranking using centrality measures [15, 34, 41], and keyphrase selection from topics detected using topic modeling [25, 40]. In this context, several extensions of PageRank and personalized PageRank have been proposed that make use of a document's citation network [12] or that bias the random walk based on the words' positions in text [10] or the words' topic distribution [27]. In order to add semantic relatedness between the words in a word graph, Martinez-Romo et al. [31] used information from WordNet.

The best performing SemEval 2010 system used term frequency thresholds to filter out phrases that are unlikely to be keyphrases, where the thresholds were estimated from the data [9]. The candidate phrases were ranked using the *tf-idf* model in conjunction with a boosting factor which aims at reducing the bias towards single word terms. Danesh et al. [8] computed an initial weight for each phrase based on a combination of the *tf-idf* score and the first position of a phrase in a document. Phrases and their initial weights were then incorporated into a graph-based algorithm, which produces the final ranking of keyphrases. Adar and Datta [1] extracted keyphrases by mining abbreviations from scientific literature and built a semantic hierarchical keyphrase database. Many of the above approaches, both supervised and unsupervised, are compared and analyzed in a survey by Hasan and Ng [17].

Neural networks have started to be incorporated into models for keyphrase extraction. For example, Wang et al. [42] investigate word embeddings to measure the relatedness between words in graph-based models. A Recurrent Neural Network (RNN) based approach is proposed by Zhang et al. [45] to identify keyphrases in Twitter data. The model addresses the problem as sequence labeling for very short text, where a joint-layer RNN is used to capture the semantic dependencies in the input sequence, but does not address the dependencies in the labels. In our work, we capture the flow of scientific writing and the dependencies between keyphrases and the other words in the text that may not necessarily exist between hashtags and the text of the tweets. Augenstein and Søgaard [2] treated keyphrase extraction as a multi-task learning problem and applied RNNs to classify keyphrase boundaries. Inspired from work in machine translation, Meng et al. [33] focused on keyphrase generation (rather than keyphrase extraction) and addressed it as a sequence to sequence learning problem with a copying mechanism, where the sequence of words in a document is used to generate a sequence of keyphrases. An Encoder-Decoder RNN, originally proposed by Cho et al. [7], was used to generate the keyphrase sequences. A variation of the model, which performs better than the Encoder-Decoder RNN, includes a copying mechanism to identify keyphrases that occur rarely in the text. Unlike Meng et al. [33], we focus on keyphrase extraction, i.e., extracting only words that are present in text, and not keyphrase generation, which outputs words that may or may not be present in text. We use the Encoder-Decoder RNN with the copying mechanism as one of our baselines.

Sequence labeling models for keyphrase extraction have shown promising results in recent studies [4, 13, 44]. For example, Gollapalli et al. [13] trained Conditional Random Fields (CRFs) to extract keyphrases from scholarly documents, using features such as *tf-idf*

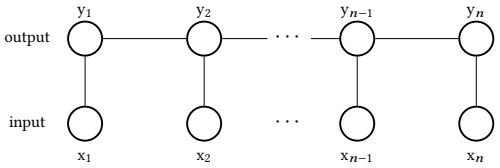


Figure 2: Layers in a CRF network.

and POS tags to predict a label for each token position in a document as being a keyphrase token (**KP**) or not (**Non-KP**). This CRF model is able to capture the dependencies in previous and future tags in the label sequence, however, the semantic dependencies in the input sequence, i.e., the text, are not incorporated.

Recently, a sequence labeling framework that takes into consideration both types of dependencies has been explored on a variety of NLP tasks such as part-of-speech tagging, noun phrase chunking, and named entity recognition [19, 26, 30]. This framework combines a bidirectional LSTM (Bi-LSTM) network as the first layer to capture sequential text dependencies with a second CRF layer to capture label dependencies. In our work, we identify the limitations of each independent model (Bi-LSTM and CRF) and explore their combination on keyphrase extraction from scholarly documents. To our knowledge, in the context of keyphrase extraction from scholarly documents, we are the first to simultaneously capture the semantics of document contexts and long-range dependencies in text by modeling the sequential text data as well as the dependencies among the labels of neighboring words.

3 METHODOLOGY

In this section, we formulate keyphrase extraction as a sequence labeling task and present the details of the Bi-LSTM-CRF model and its constituent components, CRF and Bi-LSTM.

3.1 Problem Formulation

Keyphrase extraction can be formulated as a sequence labeling task as follows: Given an input sequence $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, where each $\mathbf{x}_i$ represents the input vector of the i^{th} word, predict a sequence of labels $\mathbf{y} = \{y_1, \cdots, y_n\}$, one label for each word in the input, where each label y_i is **KP** (keyphrase word) or **Non-KP** (not keyphrase word). The sequence labeling formulation of our task takes into account the correlations between neighboring labels and allows to *jointly* decode the best sequence of labels for the input sequence, rather than decoding each label independently.

3.2 Conditional Random Fields

Conditional Random Field (CRF) introduced by Lafferty et al. [24] has been successfully used in many sequence labeling tasks and we use it here to jointly model the sequence of labels for our keyphrase extraction task. The conditional probability distribution over the label sequence $\mathbf{y}$ given $\mathbf{x}$ defined by CRF has the form:

$$p(\mathbf{y}|\mathbf{x}; \mathbf{W}, \mathbf{b}) \propto \exp\left(\sum_{i=1}^{n} \mathbf{W}^T_{y_{i-1}, y_i} \mathbf{x}_i + \mathbf{b}_{y_{i-1}, y_i}\right)$$

where $\mathbf{W}_{y_{i-1}, y_i}$ and $\mathbf{b}_{y_{i-1}, y_i}$ are model parameters (weight vector and bias) corresponding to the neighboring labels (y_{i-1}, y_i).

For training a CRF model, we estimate model parameters $\mathbf{W}$ and $\mathbf{b}$ from a training dataset $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})\}_{j=1}^{N}$ by maximizing the log-likelihood given by:

$$L(\mathbf{W}, \mathbf{b}) = \sum_{j=1}^{N} \log p(\mathbf{y}^{(j)}|\mathbf{x}^{(j)}; \mathbf{W}, \mathbf{b})$$

To find the best sequence path during decoding, the optimal sequence $\mathbf{y}$ that maximizes the likelihood, is computed using the Viterbi decoding:

$$\mathbf{y}^* = \text{argmax}_{y \in \mathcal{Y}(\mathbf{x})} p(\mathbf{y}|\mathbf{x}; \mathbf{W}, \mathbf{b})$$

Figure 2 depicts the layout of a simple CRF network. As shown in the figure, nodes in the output layer are connected, enabling the model to capture dependencies in the sequence of labels.

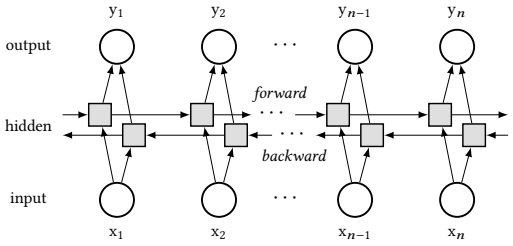


Figure 3: Layers in a Bi-LSTM network.

3.3 Bi-LSTM

Long Short Term Memory networks (LSTMs) are a special type of Recurrent Neural Networks (RNNs) and are designed to overcome the gradient vanishing problem of RNNs. In particular, LSTMs have additional memory cells, which store memory from long distance terms [18]. Because LSTMs are capable of preserving information over previous inputs of a sequence into the current input state, they have been a natural choice for applications involving temporal and sequence data such as speech recognition, language modeling and translation [14, 35]. The structure of an LSTM includes an input layer, a hidden layer and an output layer. An LSTM unit at time t consists of sub-unit-inputs ($\mathbf{i}_t$), output ($\mathbf{o}_t$), forget gates ($\mathbf{f}_t$) and memory cell ($\mathbf{c}_t$). The updates at time t are then:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^{(i)}\mathbf{h}_{t-1} + \mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)}) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^{(f)}\mathbf{h}_{t-1} + \mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^{(o)}\mathbf{h}_{t-1} + \mathbf{U}^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)}) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}^{(c)}\mathbf{h}_{t-1} + \mathbf{U}^{(c)}\mathbf{x}_t + \mathbf{b}^{(c)}) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{(t-1)} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

Here, σ is the element-wise logistic sigmoid function and $\odot$ is the Hadamard product; $\mathbf{x}_t$ is the input vector at time t, i.e., the word embedding in our case, and $\mathbf{h}_t$ is the hidden state that stores sequential information up to time t. $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are model parameters to be estimated during training.

In a forward LSTM network, the hidden state $\mathbf{h}_t$ stores information only from the past. To capture the flow of information both ways, we use a bi-directional LSTM network, which consists of a forward hidden layer and a backward hidden layer. Figure 3 shows the structure of a Bi-LSTM network. Here, the nodes in the hidden layer are connected which is how long distance information is maintained in the matrix weights.

Table 1: Dataset statistics.

Statistics	Training	Validation	Testing		
	kp527k	**kp20k-v**	**kp20k**	**WWW**	**KDD**
Number of documents	527,830	20,000	20,000	1,330	755
Number of sentences	4,686,986	176,930	177,278	12,288	7,768
Number of tokens in total	78,441,075	2,948,609	2,971,668	200,704	132,728
Number of tokens in keyphrases	5,458,743	205,586	207,073	12,181	6,119
Number of keyphrases	2,806,381	106,181	105,523	6,405	3,093

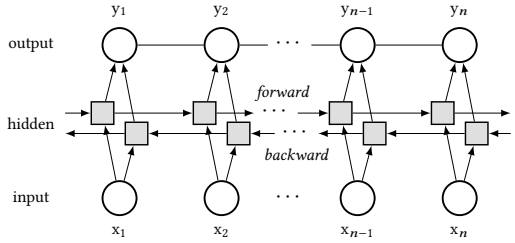


Figure 4: Layers in a Bi-LSTM-CRF network.

3.4 Bi-LSTM-CRF

In order to build a sequence labeling model that incorporates long distance information over a sequence of input as well as information on the output sequence, we combine a Bi-LSTM network with a CRF network. The network architecture is shown in Figure 4. As shown in the figure, the first layer of the model is a Bi-LSTM network with the purpose of capturing the semantics of the input text sequence. The output of the Bi-LSTM layer is passed to a CRF layer that produces a probability distribution over the tag sequence using the dependencies among labels of the entire sequence. In order to find the best sequence of labels for an input sequence, the Viterbi algorithm is used.

4 EVALUATION SETTING

To evaluate the performance of our proposed model, we conduct a wide range of experiments. In this section, we describe the datasets used for training and evaluation, discuss hyper-parameters, baselines and the evaluation measures.

4.1 Datasets

We use three different datasets of scientific documents for our evaluation purpose. The first dataset was made available by Meng et al. [33] and was collected by crawling the metadata of papers from several online digital libraries such as ACM, WebofScience, and Wiley. The dataset contains metadata for 567, 830 papers with a clear split as train, validation, and test sets provided by the authors, as follows: 527, 830 were used for model training, 20, 000 were used for parameter tuning, and the remaining 20, 000 were used for model evaluation. We refer to these sets as **kp527k**, **kp20k-validation** (or **kp20k-v**), and **kp20k-test** (or simply **kp20k**), respectively.

The second and third datasets were made available by Gollapalli and Caragea [12] and were compiled from the CiteSeerX digital library. These datasets contain metadata for research papers from the proceedings of two top-tier data mining and machine learning conferences, i.e., the ACM Conference on Knowledge Discovery and Data Mining (referred as **KDD**) and the World Wide Web Conference (referred as **WWW**).

The metadata of each paper from all three datasets above consist of titles, abstracts, and author-assigned keyphrases. The title and abstract of each paper are used to extract keyphrases, whereas the author-input keyphrases are used as gold-standard for evaluation. In experiments, for all models, we use **kp527k** for model training, **kp20k-validation** for parameter tuning, and **kp20k**, **KDD**, and **WWW** as three independent test sets for model evaluation. Note that we removed from **KDD** and **WWW** the entries that occur also in **kp527k-train** and **kp20k-validation** to avoid the overlap between train, validation, and test sets. Table **??** shows the statistics of these datasets.

4.2 Implementation Details

Since we use a sequence labeling formulation of the keyphrase extraction problem, the abstract/keyphrases data pairs are converted such that each document is a sequence of word tokens, each with a positive (**KP**) label if it occurs in a keyphrase, or with a negative (**Non-KP**) label, otherwise (consistent with [13]). We train four network models: Bi-LSTM-CRF, CRF, Bi-LSTM, and LSTM. Each sentence from a document is passed to a network as the input sequence. To evaluate performance and runtime differences, we also train Bi-LSTM-CRF models with the entire document as the input sequence (instead of each sentence at a time).

We use word embeddings as input to the above four models. The word embeddings are initialized with 100-dimension Glove pre-trained embedding vectors [37]. For all models, we use a single 100-dimension hidden layer. The LSTM, Bi-LSTM, and Bi-LSTM-CRF are optimized during training using stochastic gradient descent with learning rate $\eta_t = \frac{\eta_0}{1+\rho_t}$, where initial learning rate $\eta_0 = 10^{-2}$ and decay ratio $\rho_t = 0.5$. Gradient clipping of 5.0 is used to prevent the gradient from overflows during back-propagation. In addition, we use dropout to avoid over-fitting. We select the model with the best F1 score on the validation set, **kp20k-validation**.

Our implementation of the network models is based on a modified version of the implementation developed by [26].[1] Publicly available implementations were used for the other baselines [33].[2]

4.3 Baselines and Evaluation Measures

We contrast the Bi-LSTM-CRF network with three baselines: CRF, forward LSTM, and Bi-LSTM, and several previous models: copy-RNN [33], KEA [11], *Tf-Idf*, TextRank [34] and SingleRank [41].

[1] https://github.com/LiyuanLucasLiu/LM-LSTM-CRF
[2] https://github.com/memray/seq2seq-keyphrase

Table 2: Results (Pr, Re, F1) of Bi-LSTM-CRF in an ablation experiment on the kp20k, WWW, and KDD datasets.

Method	kp20k			WWW			KDD		
	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%
Bi-LSTM-CRF	64.19	24.66	**35.63**	64.33	28.43	**39.43**	**57.83**	31.85	**41.08**
CRF	**66.67**	10.04	17.46	**64.89**	22.11	32.98	55.76	18.69	27.99
Bi-LSTM	9.41	76.24	16.75	9.53	**76.76**	16.96	8.15	**75.93**	14.71
LSTM	9.41	**78.43**	16.81	9.51	75.71	16.90	8.13	75.38	14.68

Consistent with previous works, we evaluate the predictions of each model against the author-input keyphrases, which are available with each document, i.e., the gold standard, and report Precision, Recall, and F1-score. For model comparison in the next section, we focus the discussion of our results in terms of the F1-score, which is the harmonic mean of Precision and Recall.

5 RESULTS AND OBSERVATIONS

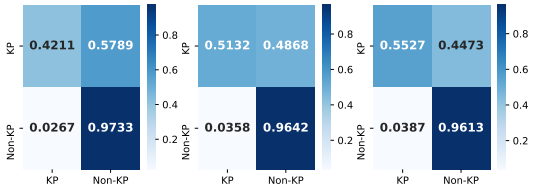
We now describe the evaluation performance of the Bi-LSTM-CRF model for extracting keyphrases from research papers.

5.1 Performance of Bi-LSTM-CRF for Keyphrase Extraction

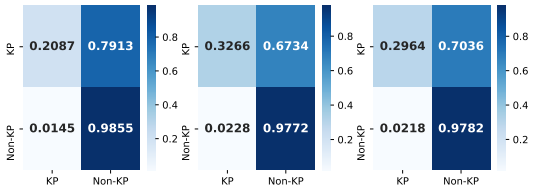
First, we evaluate the performance of Bi-LSTM-CRF in an ablation experiment to determine the role played by each component in extracting keyphrases from research papers. Specifically, we compare the Bi-LSTM-CRF model with a bidirectional LSTM, a forward LSTM, and a CRF model (by removing one component at a time from the full Bi-LSTM-CRF model).

Table 2 shows the results of this comparison. As can be seen from the table, Bi-LSTM-CRF consistently achieves the best results on all three datasets in terms of the F1-score. For example, on the **kp20k** dataset, Bi-LSTM-CRF achieves an F1-score of 35.63%, whereas CRF and Bi-LSTM achieve an F1-score of 17.46% and 16.75%, respectively. Interestingly, removing the Bi-LSTM component and using only a CRF model, we notice a steep drop in recall on all three datasets. For example, on **kp20k**, Bi-LSTM-CRF achieves a recall of 24.66%, whereas CRF alone achieves a recall of only 10.04%. This result demonstrates that leveraging the long distance semantic dependencies from text through the Bi-LSTM component of the full model is beneficial for correctly extracting a larger fraction of gold keyphrases. The CRF model alone generally achieves a similar or small increase in precision compared with the Bi-LSTM-CRF model. For example, on **WWW**, the precision values of Bi-LSTM-CRF and CRF are 64.33% and 64.89%, respectively, whereas on the **kp20k** dataset, these values are 64.19% and 66.67%, respectively.

On the other hand, removing the CRF component and using only an LSTM model (either bidirectional or forward LSTM) yields a consistent and substantial improvement in recall on all datasets over both CRF and Bi-LSTM-CRF at the expense of a dramatic drop in precision, consistently across all datasets (see Table 2). That is, on **kp20k**, the precision drops from 64.19% (achieved by Bi-LSTM-CRF) to an unacceptable value of 9.41% (obtained by LSTM alone), whereas the recall increases from 24.66% to 78.43%, obtained by Bi-LSTM-CRF and LSTM, respectively. This result indicates that LSTM is powerful in capturing the deep semantics of the text, and is



(a) Bi-LSTM-CRF



(b) CRF

Figure 5: Keyphrase extraction confusion matrices on kp20k (left), WWW (middle), and KDD (right) (results reported at word-level). The darker the blue on the main diagonal, the more accurate the model is.

able to achieve a high recall. This result also indicates that the CRF component of the full Bi-LSTM-CRF model successfully captures the label dependencies among the output labels for identifying keyphrases and contributes the most towards the full model's precision. Thus, exploiting dependencies in both the textual content of a document and the sequence of labels through the combination of an LSTM model with a CRF model yields improved results over CRF and LSTM models alone, quantified by a much larger F1-score.

Table 2 shows also that the performance of bidirectional-LSTM is very similar to that of forward LSTM. Intuitively, this makes sense since, in scientific papers, often dependencies occur in a forward fashion, e.g., patterns such as "we propose/study/explore/describe" precede keyphrases.

Moreover, from Table 2, we can also see that the performance on the **WWW** and **KDD** datasets is higher than that on the **kp20k** dataset. The lower performance on **kp20k** could be due to a larger spectrum of venues, author writing styles and author keyphrase annotations, whereas **WWW** and **KDD** are specialized datasets of papers from the data mining and machine learning communities.

Figures 5a and 5b show the confusion matrices of Bi-LSTM-CRF and CRF, on all three datasets. Each matrix is represented as a heat map, i.e., the darker the color, the higher the value at that position. An accuracy of 100% will be represented by a matrix with dark blue blocks on the main diagonal and white blocks off diagonal. As can be seen from the figures, the Bi-LSTM-CRF model that incorporates

Table 3: Comparison of Bi-LSTM-CRF with previous approaches on kp20k, WWW, and KDD datasets.

Method	kp20k			WWW			KDD		
	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%
Bi-LSTM-CRF	**64.19**	24.66	**35.63**	**64.33**	**28.43**	**39.43**	**57.83**	**31.85**	**41.08**
copyRNN @5	27.71	**41.79**	33.29	11.47	14.72	12.89	8.59	11.8	9.94
Tf-Idf @5	8.97	13.49	10.77	8.90	10.00	9.40	8.30	10.20	9.20
TextRank @5	15.29	23.01	18.37	5.80	7.10	6.20	5.10	6.50	5.60
SingleRank @5	8.42	12.70	10.14	8.80	10.90	9.50	7.70	10.30	8.60
KEA	15.14	22.78	18.19	13.57	15.25	13.86	11.39	14.50	12.42

dependencies in both the input and output, is more accurate (and thus, has a lower percentage of mis-labeled keyphrases) showed by darker cells corresponding to the **KP-KP** entry in all matrices.

5.2 Baseline Comparisons

Second, we compare the performance of Bi-LSTM-CRF with existing state-of-the-art models including supervised, unsupervised as well as deep learning models. The unsupervised models are Tf-Idf, TextRank [34], and SingleRank [41]. The supervised model is KEA[3] [43], and the deep learning model is the recently proposed copyRNN[4] [33], which is a sequence to sequence learning model based on an RNN Encoder-Decoder framework [7], combined with a copying mechanism. The Bi-LSTM-CRF, copyRNN, and KEA models are all trained on the **kp527k** dataset. For the unsupervised models and the sequence to sequence learning model, we report the performance at top 5 predicted keyphrases since top-5 showed highest performance in previous works for these models.

Table 3 shows the results of this comparison. The Bi-LSTM-CRF model outperforms all baselines in terms of the F1-score, on all three datasets. More notably, Bi-LSTM-CRF outperforms the copyRNN model in precision on all datasets, yet is slightly worse in the recall score on the **kp20k** dataset. For example, on the **kp20k**, Bi-LSTM-CRF achieves an F1-score of 35.63% as compared with 33.29% achieved by copyRNN. The precision increases from 27.71% (obtained by copyRNN) to 64.19% (obtained by Bi-LSTM-CRF) at the expense of a drop in recall from 41.79% (obtained by copyRNN) to 24.66% (obtained by Bi-LSTM-CRF).

This result is consistent with our findings in the previous section regarding the role of CRF in improving precision. Although both models use variants of RNNs to capture the semantics of the input sequence, by integrating the learning of phrasal structure into the model itself, via the CRF layer, we get higher performance than applying beam search to a decoded sequence after the learning phase, as is done in the copyRNN model.

5.3 Sentence vs. Document Level Input

Third, we conduct an experiment to examine the effect of the type of input sequence used to the models. Specifically, we compare the model where the input sequence consists of the entire document with the model where the input sequence consists of each sentence.

[3]http://www.nzdl.org/Kea/Download/Kea-4.0.zip
[4]https://github.com/memray/seq2seq-keyphrase

Table 4: Comparison results for document ("doc") vs. sentence ("sent") level for Bi-LSTM-CRF on the kp20k dataset.

Method		Pr%	Re%	F1%
Bi-LSTM-CRF	doc	**67.30**	**30.32**	**41.81**
	sent	64.19	24.66	35.63

Table 4 shows the results of this comparison for document vs. sentence level Bi-LSTM-CRF, on the **kp20k** dataset. As can be seen from the table, the performance of the Bi-LSTM-CRF that takes as input the entire content (all sentences at once) is higher than that of the Bi-LSTM-CRF that takes as input each sentence at a time. Precisely, precision, recall, and F1 values for the document-level Bi-LSTM-CRF show improvement by up to 6% over the sentence-level Bi-LSTM-CRF. This is likely due to the increased information captured by the LSTM layer (through the larger context) when using the entire content as opposed to each sentence.

A common drawback with increasing input sequence length for a recurrent neural network is that computing the gradients during back-propagation takes longer, which in turn increases the time it takes to train the model. For example, in our setting, training the Bi-LSTM-CRF model on the **kp527k** dataset at the document level took on average 22.1 hours for only five epochs, where the average input sequence size is 147 tokens, whereas when training Bi-LSTM-CRF using the sentence-level, it took only 2.7 hours, where the average number of tokens in a sentence is 16.

6 CONCLUSIONS

In this paper, we formulated the keyphrase extraction task as sequence labeling and proposed to use a Bi-LSTM-CRF model that incorporates dependencies among both the input and output sequences. To our knowledge, this is the first work to propose a neural network based sequence labeling model for keyphrase extraction from scholarly documents. The model takes advantage of the ability of Bi-LSTM to capture long distance semantic information from the input sequence and the ability of CRF to capture dependencies from the label sequence. Experimental results on three datasets showed that the proposed Bi-LSTM-CRF model that takes into account both of these dependencies play an important role in the keyphrase extraction performance.

As future work, it would be interesting to integrate the relationships between documents such as those available from a citation network. Another interesting direction would be to extend keyphrase extraction approaches to other scholarly domains.

ACKNOWLEDGEMENTS

We thank NSF for support from grants IIS-1802358, IIS-1813571, CNS-1853919, and CNS-1823288, which supported this research.

REFERENCES

[1] Eytan Adar and Srayan Datta. 2015. Building a Scientific Concept Hierarchy Database (SCHBase). In *Proceedings of the Association for Computational Linguistics*. 606–615.

[2] Isabelle Augenstein and Anders Søgaard. 2017. Multi-Task Learning of Keyphrase Boundary Classification. In *Proceedings of ACL (Volume 2: Short Papers)*, Vol. 2. 341–346.

[3] Ken Barker and Nadia Cornacchia. 2000. Using noun se heads to extract document keyphrases. (2000), 40–52.

[4] Pinaki Bhaskar, Kishorjit Nongmeikapam, and Sivaji Bandyopadhyay. 2012. Keyphrase Extraction in Scientific Articles: A Supervised Approach. In *Proceedings of COLING 2012: Demonstration Papers*. Mumbai, India, 17–24.

[5] Florin Adrian Bulgarov and Cornelia Caragea. 2015. A Comparison of Supervised Keyphrase Extraction Models. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. 13–14.

[6] Cornelia Caragea, Florin Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach. In *Proceedings of EMNLP*. 1435–1446.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Soheil Danesh, Tamara Sumner, and James H Martin. 2015. SGRank: Combining Statistical and Graphical Methods to Improve the State of the Art in Unsupervised Keyphrase Extraction. *Lexical and Computational Semantics* (2015), 117.

[9] Samhaa R El-Beltagy and Ahmed Rafea. 2010. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, 190–193.

[10] Corina Florescu and Cornelia Caragea. 2017. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, 1105–1115.

[11] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-Specific Keyphrase Extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. 668–673.

[12] Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the 28th AAAI*. 1629–1635.

[13] Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating Expert Knowledge into Keyphrase Extraction. In *Proceedings of the Thirty-First AAAI Conference, San Francisco, California, USA*. 3180–3187.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.

[15] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web*. 661–670.

[16] Khaled M Hammouda, Diego N Matute, and Mohamed S Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition*. Springer, 265–274.

[17] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the 52nd ACL*. Baltimore, Maryland, 1262–1273.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[19] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[20] Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 216–223.

[21] Steve Jones and Mark S. Staveley. 1999. Phrasier: A System for Interactive Document Retrieval Using Keyphrases. In *Proc. of the 22nd Annual International ACM SIGIR*. 160–167.

[22] Daniel Kelleher and Saturnino Luz. 2005. Automatic hypertext keyphrase detection. In *IJCAI*, Vol. 5. 1608–1609.

[23] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language resources and evaluation* 47, 3 (2013), 723–742.

[24] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[25] Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of ACL*. 620–628.

[26] Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower Sequence Labeling with Task-Aware Neural Language Model. In *Proceedings of AAAI*.

[27] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*. 366–376.

[28] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*. 257–266.

[29] Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 248–251.

[30] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, 1064–1074. http://www.aclweb.org/anthology/P16-1101

[31] Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. 2016. SemGraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science and Technology* 67, 1 (2016), 71–82.

[32] Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. ACL, 1318–1327.

[33] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. In *Proceedings of the ACL*. 582–592.

[34] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. 404–411.

[35] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[36] Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries*. Springer, 317–326.

[37] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. 1532–1543.

[38] Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. 2010. Citation Summarization Through Keyphrase Extraction. In *Proceedings of the 23rd COLING (COLING '10)*. 895–903.

[39] Lucas Sterckx, Cornelia Caragea, Thomas Demeester, and Chris Develder. 2016. Supervised Keyphrase Extraction as Positive Unlabeled Learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1924–1929.

[40] Nedelina Teneva and Weiwei Cheng. 2017. Salience Rank: Efficient Keyphrase Extraction with Topic Modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vol. 2. 530–535.

[41] Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of the 2008 AAAI*. 855–860.

[42] Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors. In *Software Engineering Research Conference*. 39.

[43] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*. ACM, 254–255.

[44] Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, and Bo Wang. 2008. Automatic Keyword Extraction from Documents Using Conditional Random Fields. *Journal of Computational Information Systems* 4(3) (2008), 1169–1180.

[45] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on Twitter. In *EMNLP*. 836–845.

[46] Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *Journal of Digital Information Management* 5, 5 (2007), 323.