Select, Extract and Generate: Neural Keyphrase Generation with Syntactic Guidance

Wasi Uddin Ahmad
University of California, Los Angeles
wasiahmad@ucla.edu

Xiao Bai
Yahoo Research
xbai@verizonmedia.com

Soomin Lee
Yahoo Research
soominl@verizonmedia.com

Kai-Wei Chang
University of California, Los Angeles
kwchang.cs@ucla.edu

ABSTRACT

Generating a set of keyphrases that summarizes the core ideas discussed in a document has a significant impact on many applications, including document understanding, retrieval, advertising, and more. In recent years, deep neural sequence-to-sequence framework has demonstrated promising results in keyphrase generation. However, processing long documents using such deep neural networks requires high computational resources. To reduce the computational cost, the documents are typically truncated before given as inputs. As a result, the models may miss essential points conveyed in a document. Moreover, most of the existing methods are either extractive (identify important phrases from the document) or generative (generate phrases word by word), and hence they do not benefit from the advantages of both modeling techniques. To address these challenges, we propose *SEG-Net*, a neural keyphrase generation model that is composed of two major components, (1) a selector that selects the salient sentences in a document, and (2) an extractor-generator that jointly extracts and generates keyphrases from the selected sentences. SEG-Net uses a self-attentive architecture, known as, *Transformer* as the building block with a couple of uniqueness. First, SEG-Net incorporates a novel *layer-wise* coverage attention to summarize most of the points discussed in the target document. Second, it uses an *informed* copy attention mechanism to encourage focusing on different segments of the document during keyphrase extraction and generation. Besides, SEG-Net jointly learns keyphrase generation and their part-of-speech tag prediction, where the later provides syntactic supervision to the former. The experimental results on seven keyphrase generation benchmarks from scientific and web documents demonstrate that SEG-Net outperforms the state-of-the-art neural generative methods by a large margin in both domains.

KEYWORDS

Neural keyphrase generation, sentence classification, multi-task learning, part-of-speech tagging, Transformer

1 INTRODUCTION

Keyphrases are short pieces of text that summarize the key points discussed in a document. They are useful for many natural language processing and information retrieval tasks, such as, text summarization [55, 57], question answering [41, 44], sentiment analysis [2, 49], document retrieval [21, 24], document categorization or clustering [16, 19], contextual advertisement [8, 51], and more. In the automatic keyphrase generation task, the input is a document, and the output is a set of keyphrases that can be categorized as *present* or *absent* keyphrases. Present keyphrases appear exactly in the target document, while absent keyphrases are only semantically related and have partial or no overlap to the target document. We provide an example of a target document and its keyphrases in Figure 1.

Title: [1] natural language processing technologies for developing a language learning environment .

Abstract: [1] so far , computer assisted language learning (call) comes in many different flavors . [1] our research work focuses on developing an integrated e learning environment that allows improving language skills in specific contexts . [1] integrated e learning environment means that it is a web based solution... , for instance , web browsers or email clients . [0] it should be accessible... [1] natural language processing (nlp) forms the technological basis for developing such a learning framework . [0] the paper gives an overview... [0] therefore , on the one hand , it explains creation... [0] on the other hand , it describes existing nlp standards . [0] based on our requirements , the paper gives... [1] an outlook at the end points out necessary developments in e learning to keep in mind .

Present keyphrases: natural language processing; computer assisted language learning; integrated e learning

Absent keyphrases: semantic web technologies; learning of foreign languages

Predictions of our proposed model, SEG-Net

Salience labels: 1 1 1 1 0 1 0 1 0 0 1

Present keyphrases: natural language processing; computer assisted language learning

Absent keyphrases: web based learning; learning natural language

Figure 1: Sample document with gold keyphrase labels and our model's predictions. The numeric value in brackets [ ] represent *salience* label (0/1) indicating whether the sentence contains any present keyphrase or overlaps with any absent keyphrase.

Automatic keyphrase generation methods in literature can be broadly divided into *extraction* and *generation* methods. A large pool of prior works have been devoted to extracting keyphrases by selecting text spans or phrases directly from the target document (e.g., "natural language processing" in Figure 1) [18, 22, 29, 31, 33, 48, 54]. However, due to their design principle, these approaches cannot predict absent keyphrases. In recent years, the neural sequence-to-sequence (Seq2Seq) framework [42] has become the fundamental building block in neural keyphrase generation models with its widespread usage in natural language generation tasks. The first deep neural keyphrase generation model, CopyRNN [35] adopts the Seq2Seq framework [1, 32] with copy mechanism [15, 39]. With the copy attention mechanism, the Seq2Seq models are capable of generating both present and absent keyphrases. A few subsequent works [4, 6, 53] extended CopyRNN to enhance keyphrase generation. Although these generative approaches are capable of generating both present and absent phrases, they ignore the advantages of the extractive solutions, e.g., extracted keyphrases indicate the essential segments of the target document.

To generate a comprehensive set of keyphrases that summarizes the key points conveyed in the target document, reading the full document content is necessary. However, to the best of our knowledge, none of the previous neural methods are provisioned to read the full content of a document as it can be thousands of words long (e.g., news articles). Processing such long documents through deep neural networks requires high computational resources. Hence, the existing neural methods truncate the target document; take the first few hundred words as input and ignore the rest of the document that may contain salient information.

To address the aforementioned challenges, in this paper, we propose SEG-Net (stands for **S**elect, **E**xtract, and **G**enerate) that has two major components, (1) a *sentence-selector* that selects the salient sentences in a document and (2) an *extractor-generator* that predicts the present keyphrases and generates the absent keyphrases jointly. The primary motivation to design the sentence-selector is to decompose a long target document into small segments, e.g., sentences, paragraphs, and identify the salient ones for keyphrase generation. For example, as shown in Figure 1, we split a document into a list of sentences and classify them with salient and non-salient labels. In this context, we consider a sentence as salient if it contains present keyphrases or overlaps with absent keyphrases. In Figure 1, the sample document consists of six salient (label 1) and five non-salient sentences (label 0). A similar notion is adopted in prior works on text summarization [7, 28] and question answering [36].

We employ *Transformer* [46] as the backbone of the extractor-generator in SEG-Net. We chose Transformer as it completely relies on a self-attention mechanism that is capable of capturing longer range dependencies. We equip the extractor-generator with a novel *layer-wise* coverage attention and an *informed* copy attention such that the generated keyphrases summarize the entire target document. The layer-wise coverage attention keeps track of the target document segments that are covered by previously generated phrases to guide the self-attention mechanism in Transformer while attending the encoded target document in future generation steps.

We revise the standard copy mechanism [15, 39] and propose an "informed" copy attention for keyphrase generation. Our revision is based on the observation that a word has a different meaning in the context of two different keyphrases. For example, in Figure 1, the word "learning" has a different meaning when used in "computer assisted language learning", "integrated e learning", and "learning of foreign languages". Hence, we revise the copy mechanism so that SEG-Net does not copy a word from a present keyphrase while generating an absent keyphrase. Another motivation behind such a copy mechanism is to encourage the model to generate absent keyphrases that summarize the other segments of the target document that are not covered by the present keyphrases.

We train SEG-Net via multi-task learning to predict keyphrases as well as their part-of-speech (POS) tags. We exploit the multi-layer structure of Transformer to perform both POS tagging and keyphrase generation. We evaluate SEG-Net on five benchmarks from scientific articles and two benchmarks from web documents to demonstrate its effectiveness over the state-of-the-art neural generative methods on both domains. We perform thorough ablation and analysis to present noteworthy findings such as (1) selecting salient sentences significantly improve present keyphrase extraction, (2) the layer-wise coverage attention and informed copy mechanism facilitates absent keyphrase generation, (3) jointly learning POS tagging and phrase prediction reduces duplicate and overlapping keyphrase generation.

2 RELATED WORK

2.1 Automatic Keyphrase Extraction

The keyphrase extraction methods identify notable phrases that appear in a document. The existing approaches generally work in two steps. First, they select a set of candidate keyphrases from the target document. The candidate keyphrases are chosen based on heuristic rules, such as essential n-grams or noun phrases [18, 34] or phrases whose part-of-speech (POS) tags match a specific sequence [27, 29, 48]. In the second step, the selected keyphrases are scored as per their importance to summarize key points of the target document, which is computed by unsupervised ranking approaches [14, 47] or supervised learning algorithms [18, 30, 33, 37, 50]. Finally, the top-ranked candidates are returned as the keyphrases. Another type of extractive solution follows a sequence tagging approach where the likelihood of each word in the source text to be a keyphrase word is learned sequentially [11, 12, 31, 54]. However, these extractive solutions are only able to predict the keyphrases that appear in the document and thus fail to predict the absent keyphrases.

2.2 Automatic Keyphrase Generation

In contrast to the extraction methods, the generative approaches aim at predicting both the present and absent keyphrases of a target document. Meng et al. [35] proposed the first generative model, known as CopyRNN, which is composed of an encoder-decoder with attention [1, 32] and copy mechanism [15, 39]. Multiple extensions of CopyRNN were also proposed in subsequent works. Chen et al. [4] proposed CorrRNN that incorporates coverage [45] and review mechanisms. TG-Net [6] considers the influence of the title and proposed a title-guided encoding for the input document to improve keyphrase generation. However, all these generative methods are trained to predict *one* keyphrase from the target document. To generate a fixed number of keyphrases, they use a beam search and select the top-k as the final predictions. In contrast, Yuan et al. [53] proposed to concatenate all the ground-truth keyphrases and train models to generate them as one output sequence.

All these previous generative models utilize a recurrent neural network (RNN) to learn document representations and keyphrase generation word by word. Different from these approaches, Zhang et al. [56] proposed CopyCNN that utilizes convolutional neural network (CNN) [23] to form sequence-to-sequence architecture. In contrast, we use the self-attention mechanism [46] to learn document representations. Our choice is based on the finding that RNN-based approaches face difficulty in remembering long-range dependencies due to its sequential nature [9]. Another crucial difference is, the prior works use truncated documents to generate keyphrases. In contrast, our model generates keyphrases from the salient sentences to cover all the key concepts discussed in the target document.

Keyphrases that summarize a document are mostly noun phrases and commonly consist of nouns and adjectives. Recently, Zhao and Zhang [58] proposed to learn keyphrase generation and part-of-speech (POS) tagging jointly. However, they utilized two *parallel* encoder-decoder to generate the keyphrases and the corresponding POS tag sequence. In contrast, our model uses a multi-layer decoder where the hidden representations produced by the lower and upper layers are used for POS tagging and keyphrase generation, respectively. We anticipate that in this manner, POS tagging as an auxiliary task provides better syntactic supervision for keyphrase generation. Other noteworthy approaches in literature propose to integrate external knowledge and to separate keyphrase extraction from generation [5], utilizing semi-supervised learning [52] to leverage unlabeled data, applying adversarial training [43] or reinforcement learning [3] to improve keyphrase generation.

3 SEG-NET FOR KEYPHRASE GENERATION

The problem of keyphrase generation is defined as given a document x, generate a set of keyphrases $\mathcal{K} = \{k^1, k^2, \dots, k^{|\mathcal{K}|}\}$ where the document $x = [x_1, \dots, x_{|x|}]$ and each keyphrase $k^i = [k^i_1, \dots, k^i_{|k^i|}]$ is a sequence of words. A document can be split into a list of sentences, $\mathcal{S}_x = [s^1_x, s^2_x, \dots, s^{|S|}_x]$ where each sentence $s^i_x = [x_j, \dots, x_{j+|s^i|-1}]$ is a consecutive subsequence of the document x with begin index $j \leq |x|$ and end index $(j + |s^i|) < |x|$. In literature, keyphrases are categorized into two types, *present* and *absent* keyphrases. A present keyphrase is a consecutive subsequence of the document, while an absent keyphrase is not. However, an absent keyphrase may have a partial overlapping with the document's word sequence. We denote the sets of present and absent keyphrases as $\mathcal{K}_p = \{k^1_p, k^2_p, \dots, k^{|\mathcal{K}^p|}_p\}$ and $\mathcal{K}_a = \{k^1_a, k^2_a, \dots, k^{|\mathcal{K}^a|}_a\}$, respectively. Hence, we can express a set of keyphrases as $\mathcal{K} = \mathcal{K}_p \cup \mathcal{K}_a$.

SEG-Net decomposes the keyphrase generation task into three sub-tasks, (1) salient sentence selection, (2) present keyphrase extraction, and (3) absent keyphrase generation. We define them below.

TASK 1. ***(Salient Sentence Selection)*** *Given a list of sentences from a document, predict a binary label* $(0/1)$ *for each sentence. The positive label* (1) *indicates that the sentence either contains a present keyphrase or overlaps with an absent keyphrase.*

TASK 2. ***(Present Keyphrase Extraction)*** *Given a list of salient sentences as a concatenated sequence of words, predict a binary label* $(0/1)$ *for each word. The positive label* (1) *indicates that the word is a constituent of a present keyphrase.*

TASK 3. ***(Absent Keyphrase Generation)*** *Given a list of salient sentences as a concatenated sequence of words, generate a set of absent keyphrases in a sequence-to-sequence [42] fashion.*

SEG-Net uses Transformer [46] as the building block and incorporates novel layer-wise coverage and informed copy attention. In this section, first, we briefly discuss the background concepts (3.1) that will facilitate our model description. Then, we detail how SEG-Net carries out the salient sentence selection and jointly extracts and generates the present and absent keyphrases (3.2). Finally, we describe how we train SEG-Net via multi-task learning (3.3).

3.1 Background

The architecture of SEG-Net is based on Transformer [46] that employs an encoder-decoder structure, consisting of stacked encoder and decoder layers. Each encoder and decoder layer consists of two and three sublayers, respectively. An encoder layer is composed of multi-head attention, followed by a position-wise feed-forward layer. A decoder layer has two consecutive multi-head attention layers followed by a position-wise feed-forward layer. The multi-head attention employed in Transformer is based on the self-attention mechanism where each element in an input sequence attends to every other element of the input, including itself, while computing the output representation.

In the first multi-head attention sublayer, the decoder uses masking in its self-attention to prevent an element from incorporating information about future elements during training. This is analogous to the *unidirectional* RNN that is employed as a decoder in a recurrent sequence-to-sequence network. The second multi-head attention sublayer in a decoder layer applies self-attention between the encoded input sequence and each element of the decoded sequence. This attention sublayer is akin to the encoder-decoder attention in sequence-to-sequence learning [1, 32]. Each encoder and decoder layer uses residual connections followed by layer normalization around each of the sublayers. The encoder-decoder layer of the Transformer is depicted in Figure 2.

3.1.1 *Multi-head Attention.*

In multi-head attention, there are h identical attention heads that employ the self-attention mechanism. In each attention head, a sequence of input vectors, $\mathbf{x} = (x_1, \dots, x_n)$ where $x_i \in R^{d_{model}}$ are transformed into a sequence of output vectors, $\mathbf{o} = (o_1, \dots, o_n)$ where $o_i \in R^{d_k}$ as:

$$o_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V \right), \tag{1}$$

$$e_{ij} = \frac{x_i W^Q \left(x_j W^K \right)^T}{\sqrt{d_k}}; \; \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}, \tag{2}$$

where $W^Q, W^K \in R^{d_{model} \times d_k}, W^V \in R^{d_{model} \times d_v}$ are the parameters that are unique per attention head.

3.1.2 *Position Representations.*

Transformer neither employs recurrence nor convolution. Hence, we need to inject either absolute or relative position of the tokens in a sequence explicitly to make use of the order information. In the keyphrase generation task, we employ absolute and relative position representation during encoding the input document and decoding the keyphrases, respectively.

Absolute Position Representations. We train an embedding matrix W_{posi} that learns to encode tokens' absolute positions in a document into vectors of dimension d_{model}.

Relative Position Representations. Since we aim to generate a *set* of keyphrases, the order of their generation should not matter. Hence, we want the decoder to learn the pairwise relationships between the keyphrases instead of a specific ordering such as alphabetical ordering, length-based ordering, etc.

To encode the pairwise relationships between input elements, Shaw et al. [40] extended the self-attention mechanism as follows.

$$o_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V + a_{ij}^V \right),$$

$$e_{ij} = \frac{x_i W^Q \left(x_j W^K + a_{ij}^K \right)^T}{\sqrt{d_k}},$$

where, a_{ij}^V and a_{ij}^K are relative positional representations for the two position i and j. Shaw et al. [40] suggested clipping the maximum relative position to a maximum absolute value of k.

$$a_{ij}^K = w_{clip(j-i,k)}^K, \; a_{ij}^V = w_{clip(j-i,k)}^V,$$

$$clip(x, k) = \max(-k, \min(k, x)).$$

Hence, we learn $2k + 1$ relative position representations: $w^K = (w_{-k}^K, \ldots, w_k^K)$, and $w^V = (w_{-k}^V, \ldots, w_k^V)$.

3.1.3 Layer-wise Coordination. In Transformer, the stacked encoder layers generate a sequence of output vector representations for the source sequence layer by layer, gradually from the lowest layer to the highest layer. Then each layer of the decoder takes the last layer (the highest layer) representations from the encoder as inputs while computing the output representations for each position in the target sequence. In essence, all the layers in the decoder attend the output representations produced by the last layer of the encoder. This design choice makes it challenging to apply the coverage mechanism [4, 45]. Hence, we adopt the layer-wise coordination [17] between encoder-decoder in Transformer.

According to layer-wise coordination, the i-th layer of the decoder generates output representations for a target token based on $(i-1)$-th layer's output representations of that token and its preceding tokens and the output representation for all the source tokens from the i-th layer of the encoder. In simpler words, with the same number of encoder and decoder layers, the output representation produced by a decoder layer depends on the output representations provided by the corresponding encoder layer and the previous layer of the decoder. Note that the incorporation of the layer-wise coordination requires the same number of layers in encoder and decoder.

3.2 SEG-Net

Our proposed model, SEG-Net jointly learns to extract and generate present and absent keyphrases from the salient sentences in a target document. The key advantage of SEG-Net is the maximal utilization of the information from the input text in order to generate a set of keyphrases that summarize all the key points in the target document. SEG-Net consists of a *sentence-selector* and an *extractor-generator*. The sentence-selector identifies the salient sentences from the target document (Task 1) that are fed to the extractor-generator to predict both the present and absent keyphrases (Task 2, 3). The sentence-selector consists of an embedding layer, stacked Transformer encoder layers, and a classifier. The extractor-generator comprises an embedding layer, Transformer, an extractor, a copy attention layer, and a softmax layer. We briefly describe them in this section.

3.2.1 Embedding Layer. The embedding layer maps each word in an input sequence to a low-dimensional vector space. We train four embedding matrix, W_e, W_{pos}, W_{posi}, and W_{seg} that converts a

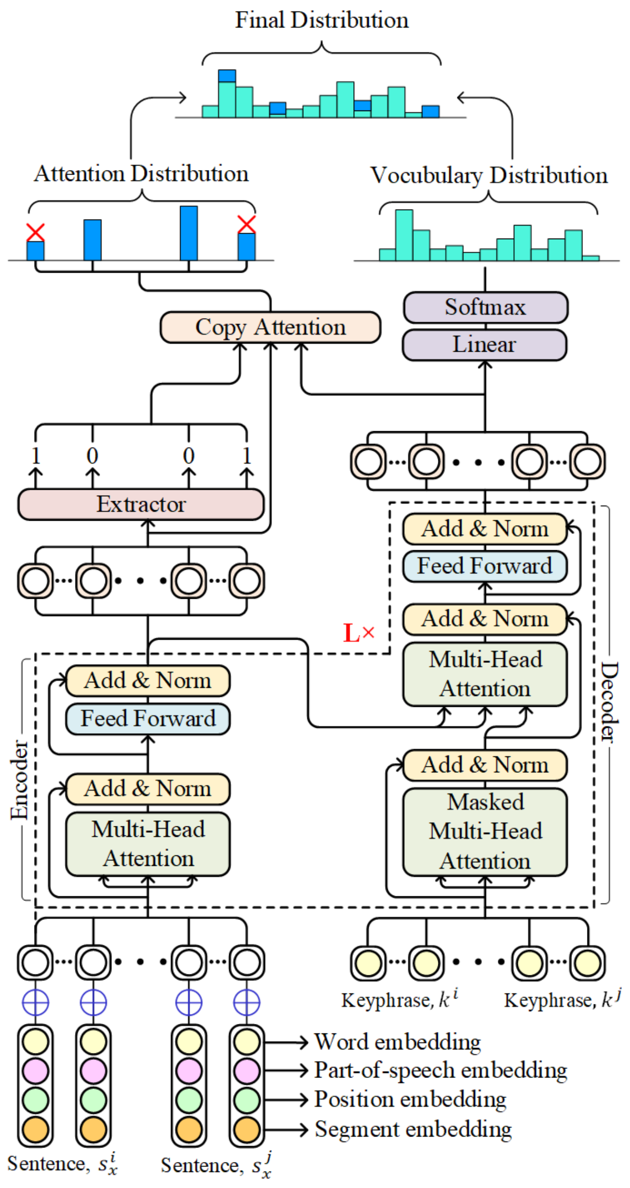

Figure 2: Overview of the extractor-generator module of our proposed SEG-Net. The three major components are encoder, extractor, and decoder. The encoder encodes the salient sentences of the input document. The extractor predicts the constituent words of the present keyphrase while the decoder generates the absent keyphrases word by word. We employ an informed copy mechanism that does not allow to copy extracted keyphrase words during absent keyphrase generation.

word, its part-of-speech tag, absolute position, and segment id into vector representations of size d_{model}. The segment id of a word indicates the index of its pertaining sentence in a document. In addition, we obtain a character-level embedding for each word[1] using

[1] We do not depict the character-level embeddings in Figure 2 for simplicity.

Convolutional Neural Networks (CNN) [23]. To learn a fixed-length vector representation of a word, we add the five embedding vectors element-wise. To form the vector representations of the keyphrase tokens, we only use their word and character-level embeddings.

3.2.2 Sentence-Selector.

The objective of the sentence-selector is to predict the salient sentences in a document, as described in Task 1. In a nutshell, the sentence-selector is a binary text classifier. Given a sentence, $s_x^i = [x_j, \ldots, x_{j+|s^i|-1}]$ from a document x, the selector predicts the salience probability of that input sentence. First, the embedding layer maps each of the words in the sentence into a d_{model} dimensional vector. The sequence of word vectors are fed to a stack of Transformer encoder layers that produce a sequence of output representations $[o_j, \ldots, o_{j+|s^i|-1}]$ where $o_t \in R^{d_{model}}$ as described in section 3.1.

Then we apply max and mean pooling on the output representations to form $s_{max}, s_{mean} \in R^{d_{model}}$ that are concatenated $s_{pool} = s_{max} \oplus s_{mean}$ to form the sentence embedding vector. We feed the sentence vector s_{pool} through a three-layer, batch-normalized [20] maxout network [13] to predict the salience probability.

3.2.3 Extractor-Generator.

The extractor-generator module takes a list of salient sentences of a document as an input that are concatenated to form a sequence of words and predicts the present and absent keyphrases. We illustrate the extractor-generator module in Figure 2 and briefly describe its major components as follows.

Encoder. The encoder contains an embedding layer (described in 3.2.1) followed by an L-layer Transformer encoder. Each word in the input sequence $[x_1, \ldots, x_n]$ is first mapped to an embedding vector. Then the sequence of word embeddings is fed to the Transformer encoder that produces a sequence of contextualized word representations $[o_1^l, \ldots, o_n^l]$ where $l = 1, \ldots, L$ using the multi-head self-attention mechanism (described in 3.1.1).

Extractor. The extractor takes the encoded word vectors $[o_1, \ldots, o_n]$ as input and predicts the probability of each word being a constituent of a present keyphrase via a sigmoid function.

$$p(\mu_j = 1|o_j^L) = \sigma\big(W_{r_2}(\tanh(W_{r_1} o_j^L + b_{r_1})) + b_{r_2}\big),$$

where $W_{r_1}, W_{r_2}, b_{r_1}, b_{r_2}$ are trainable parameters.

Decoder. The decoder generates the absent keyphrases as a concatenated sequence of words $[y_1^*, \ldots, y_m^*]$ where m is the sum of the length of the absent keyphrases. The decoder employs an embedding layer, L-layers of Transformer decoder followed by a softmax layer. The embedding layer converts the keyphrase tokens into vector representations that are fed to the Transformer decoder. The output of the last (L-th) decoder layer $h_1^L, \ldots, h_m^L$ is passed through a softmax layer to predict a probability distribution over the vocabulary V.

$$p(y_t^*|y_{1:t-1}^*, x) = softmax(W_v h_t^L + b_v), \tag{3}$$

where $W_v \in R^{|V| \times d_v}$ and $b_{word} \in R^{|V|}$ are trainable parameters.

3.2.4 Syntactic Guidance.

Most of the keyphrases are noun phrases and commonly consist of nouns and adjectives. Hence, to guide the keyphrase generation, we propose to learn keyphrase generation with part-of-speech (POS) tagging jointly. Hence, we propose to use the output of the l-th decoder layer $h_1^l, \ldots, h_m^l$ by passing through a softmax layer to predict a probability distribution over a predefined vocabulary of POS tags Ψ.

$$p(\tau_t^*|\tau_{1:t-1}^*, x) = softmax(W_{tag} h_t^l + b_{tag}),$$

where $W_{tag} \in R^{|\Psi| \times d_{model}}, b_{tag} \in R^{|\Psi|}$ are trainable parameters.

We use the output from a lower layer of the decoder for POS tagging and use the highest layer (the last layer) to generate the absent keyphrases. Since POS tagging requires learning the syntactic structure of the keyphrases, we hypothesize that it can guide the decoder to generate structurally coherent keyphrases.

3.2.5 Coverage and Copy Attention.

A set of keyphrases should summarize all the key points discussed in a target document. To achieve this goal, we incorporate a novel layer-wise coverage attention and an informed copy mechanism in the extractor-generator.

Coverage Attention. The coverage attention [4, 45, 53] keeps track of the parts in the document that has been covered by previously generated phrases and guides future generation steps such that generated keyphrases summarize the entire document. To equip the multi-layer structure of the extractor-generator with the coverage attention mechanism, we modify the encoder-decoder multi-head attention at each layer in Transformer. To this end, we modify the Eq. (2) as $\alpha_{ij} = \frac{f_{ij}}{\sum_{k=1}^{n} f_{ik}}$ and calculate f_{ij} as follows.

$$f_{ij} = \begin{cases} \exp(e_{ij}^t) & \text{if } t = 1 \\ \frac{\exp(e_{ij}^t)}{\sum_{k=1}^{t-1} \exp(e_{ij}^k)} & \text{otherwise,} \end{cases}$$

where e_{ij}^t is same as e_{ij} in Eq. (2) computed at time step t. Note that we adopt the layer-wise coordination technique (3.1.3) to facilitate the design of the layer-wise coverage attention.

Copy Attention. We incorporate an *informed* copying mechanism in the extractor-generator to facilitate absent keyphrase generation. Note that absent keyphrases have partial or no overlapping with the target document. With the copy mechanism, we want the decoder to learn to copy phrase terms that overlap with the target document. However, we do not want the decoder to copy a present keyphrase term during absent keyphrase generation, because a word has a different meaning when present in the context of two keyphrases.

Formally, we take the output from the last layer of the encoder $[o_1^L, \ldots, o_n^L]$ and compute the attention score of the decoder output h_t^L at time step t as: $att(o_i^L, h_t^L) = o_i^L W_{att} h_t^L$. Then we compute the context vector, c_t^L at time step t:

$$a_{ti}^L = \frac{att(o_i^L, h_t^L)}{\sum_{k=1}^{n} exp(att(o_k^L, h_t^L))}; \; c_t^L = \sum_{i=1}^{n} a_{ti}^L o_i^L.$$

The copy mechanism uses the attention weights a_{ti}^L as the probability distribution $p(y_t^* = x_i|u_t = 1) = a_{ti}^L$ to copy the input tokens x_i. We compute the probability of using the copy mechanism at the decoding step t as $p(u_t = 1) = \sigma(W_u[h_t^L || c_t^L] + b_u)$, where $||$ denotes the vector concatenation operator. Then we obtain the final probability distribution for the output token y_t^* as follows.

$$p(y_t^*) = p(u_t = 0)p(y_t^*|u_t = 0) + p(u_t = 1)p(y_t^*|u_t = 1),$$

where $p(y_t^*|u_t = 0)$ is defined in Eq. (3). In the above equations, all probabilities are conditioned on $y_{1:t-1}^*, x$, but we omit them to keep the notations simple. We set $p(u_t = 0) = 1$ to block copying a word from the source text if that is a constituent of a present keyphrase.

Dataset	# Example	Max / Avg. Source Len.	Max / Avg. # Sentence	% Sentence*	Max / Avg. Keyphrase Len.	Avg. # Keyphrase	% Present	% Absent
KP20k	20,000	1,438 / 179.8	108 / 7.8	29.2	23 / 2.04	5.28	62.9	37.1
Inspec	500	386 / 128.7	23 / 5.5	16.5	10 / 2.48	9.83	73.6	26.4
Krapivin	400	554 / 182.6	28 / 8.2	28.3	6 / 2.21	5.84	55.7	44.3
Nus	211	973 / 219.1	42 / 11.8	32.6	70 / 2.22	11.65	54.4	45.6
SemEval	100	473 / 234.8	22 / 11.9	27.0	11 / 2.38	14.66	42.6	57.4
KPTimes	20,000	7,569 / 777.9	631 / 28.9	35.4	18 / 1.84	5.27	58.8	41.2
In-house	26,000	9,745 / 969.1	538 / 35.6	44.0	16 / 2.69	4.08	37.5	62.5

Table 1: Summary of the test portion of the keyphrase benchmarks used in experiments. Sentence* represents the percentage of non-salient sentences in the input text. % Present and % Absent indicate the percentage of keyphrases for the respective types.

3.3 Learning Objectives

We train the two major components of SEG-Net, the sentence-selector, and the extractor-generator individually.

3.3.1 Sentence-Selector Training. For each sentence in a document x, the selector predicts the salience label. We choose the sentences containing present keyphrases or overlap with absent keyphrases as the gold salient sentences and use the weighted cross-entropy loss for selector training.

$$\mathcal{L}_s = -\frac{1}{|x|}\sum_{j=1}^{|x|} \omega\vartheta_j^* \log \vartheta_j + (1-\vartheta_j^*)\log(1-\vartheta_j), \qquad (4)$$

where $\vartheta_j^* \in \{0,1\}$ is the ground-truth label for the j-th sentence and ω is a hyper-parameter to balance the importance of positive and negative examples of salience sentences.

3.3.2 Extractor-Generator Training. The extractor-generator takes a list of salient sentences as a concatenated sequence of words and predicts the present keyphrases and generates the absent keyphrases and their part-of-speech tags. We optimize the parameters of the extractor-generator via supervised multi-task training.

Extraction Loss. For each word of the input sequence, the extractor predicts whether the word appears in a contiguous subsequence that matches a present keyphrase. The extractor treats the task as a binary classification task and we compute the extraction loss $\mathcal{L}_e$ as the weighted cross-entropy loss as in Eq. (4).

Generation Loss. The decoder jointly generates the set of absent keyphrases and their part-of-speech (POS) tags as a concatenated sequence of tokens. Hence, the total generation loss $\mathcal{L}_g$ is computed as the weighted average of the keyphrase generation loss $\mathcal{L}_w$ and part-of-speech tag generation loss $\mathcal{L}_{tag}$ as: $\mathcal{L}_g = \alpha\mathcal{L}_w + (1-\alpha)\mathcal{L}_{tag}$. We compute $\mathcal{L}_w$ as the negative log-likelihood loss of the ground-truth keyphrases generation.

$$\mathcal{L}_w = -\sum_{t=1}^{n} \log\ p(y_t^*|y_1^*, \ldots, y_{t-1}^*, x), \qquad (5)$$

where n is sum of the length of all absent keyphrases.

Similarly, we compute the negative log-likelihood loss $\mathcal{L}_{tag}$ for POS tag sequence generation.

Overall Loss. The overall loss to train the extractor-generator is a weighted average of the extraction and generation loss.

$$\mathcal{L}_{eg} = \beta\mathcal{L}_e + (1-\beta)\mathcal{L}_g.$$

4 EXPERIMENTS

4.1 Datasets and Preprocessing

Keyphrases for scientific documents. We conduct experiments on five scientific benchmarks from the computer science domain: KP20k [35], Inspec [18], Krapivin [26], NUS [37], and SemEval [22]. Each example from these datasets consists of the title, abstract, and a list of keyphrases. Following previous works [3, 5, 6, 35, 53], we use the training set of the largest dataset, KP20k, to train and employ the testing datasets from all the benchmarks to evaluate the baselines and our models. KP20k dataset consists of 530,000 and 20,000 articles for training and validation, respectively.

Keyphrases for web documents. We perform experiments on two datasets that consist of news articles and general web documents. The first dataset is KPTimes [10] that provides news text paired with editor-curated keyphrases. The second dataset is an *in-house* dataset that was generated from the click logs of a large-scale commercial web search engine. Specifically, we randomly sampled web documents that were clicked at least once during the month of February in 2019. For each sampled web document, we collected 20 queries that led to the highest number of clicks on it. This design choice is motivated by the observation that queries frequently leading to clicks on a web document usually summarize the main concepts in the document. We further filter out the less relevant queries. The relevance score for each query is assigned by an in-house query-document relevance model. We also remove duplicate queries by comparing their bag-of-words representation.[2] The dataset consists of 206,000, 24,000, and 26,000 unique web documents for training, validation, and evaluation, respectively.

Detailed statistics of the test portion of the experiment datasets are provided in Table 1. Following Meng et al. [35], we apply lowercasing, tokenization and replacing digits with $\langle digit \rangle$ symbol to preprocess all the datasets. We use *spaCy* to get the sentence boundaries and part-of-speech tags.

4.2 Baseline Models and Evaluation Metrics

For a comprehensive evaluation, we compare the performance of SEG-Net with four state-of-the-art neural generative methods, catSeq [53], catSeqD [53], catSeqCorr [4], and catSeqTG [6]. In addition, we consider a variant of catSeq model as baseline where we replace the RNN-based Seq2Seq architecture with the Transformer. The catSeq, catSeqCorr and catSeqTG models are known as CopyRNN [35],

[2]We perform stemming before computing the bag-of-words representations.

Model	KP20k		Inspec		Krapivin		NUS		SemEval	
	F1@M	F1@5	F1@M	F1@5	F1@M	F1@5	F1@M	F1@5	F1@M	F1@5
Present Keyphrase Generation										
catSeq [53]	0.367	0.291	0.262	0.225	0.354	0.269	0.397	0.323	0.283	0.242
catSeqD [53]	0.363	0.285	0.263	0.219	0.349	0.264	0.394	0.321	0.274	0.233
catSeqCorr [4]	0.365	0.289	0.269	0.227	0.349	0.265	0.390	0.319	0.290	0.246
catSeqTG [6]	0.366	0.292	0.270	0.229	0.366	0.282	0.393	0.325	0.290	0.246
catSeq (Transformer)	0.368	0.291	0.264	0.225	0.356	0.274	0.405	0.328	0.288	0.245
SEG-Net (This work)	**0.381**†	**0.323**†	**0.301**†	**0.246**†	**0.378**†	**0.299**†	**0.459**†	**0.401**†	**0.341**†	**0.298**†
Absent Keyphrase Generation										
catSeq [53]	0.032	0.015	0.008	0.004	0.036	0.018	**0.028**†	**0.016**	0.028	0.020
catSeqD [53]	0.031	0.015	0.011	0.006	0.037	0.018	0.024	0.015	0.024	0.016
catSeqCorr [4]	0.032	0.015	0.009	0.005	0.038	0.020	0.024	0.014	0.026	0.018
catSeqTG [6]	0.032	0.015	0.011	0.005	0.034	0.018	0.018	0.011	0.027	0.019
catSeq (Transformer)	0.031	0.015	0.009	0.005	0.038	0.020	**0.028**†	**0.016**	0.029	0.020
SEG-Net (This work)	**0.038**†	**0.020**†	**0.014**†	**0.009**†	**0.056**†	**0.028**†	0.024	0.014	**0.031**	**0.021**

Table 2: Results of keyphrase prediction on the five scientific benchmarks. The bold-faced values indicate the best performances across the board. † means the model is statistically significantly better (by paired bootstrap test, p < 0.05) than all other models.

CorrRNN [4] and TGNet [6] respectively. CopyRNN, CorrRNN or TGNet generates one keyphrase in a sequence-to-sequence fashion and use beam search to generate multiple keyphrases. In contrast, we follow [53] and concatenate all the keyphrases into one output sequence using a special delimiter "<sep>". We used the publicly available implementation of these baselines[3] in our experiment.

To measure the accuracy of the sentence-selector, we use micro averaged F1 score. We also compute precision and recall to compare the performance of the sentence-selector with a baseline. While in SEG-Net, we select up to N predicted salient sentences, in the baseline method, the first N sentences are selected from the target document so that their total length does not exceed a predefined word limit. In keyphrase generation, the accuracy is typically computed by comparing the top k predicted keyphrases with the ground-truth keyphrases. Most previous works [3–5, 35, 52] used F1@k where the evaluation cutoff k is set to 5, 10, or M where M represents the number of model predictions. In this paper, we report F1@M and F1@5 for all the baselines and our models. Following Chan et al. [3], we append random wrong answers to the predictions if a model generates less than five keyphrases. We use marco average to aggregate the evaluation scores for all testing samples. We apply Porter Stemmer before determining whether two keyphrases are identical and remove all the duplicated keyphrases from the predictions before computing an evaluation score.

4.3 Implementation Details

Hyper-parameters. We use a fixed vocabulary of the most frequent $|V| = 50,000$ words in both sentence-selector and extractor-generator. We truncate the target document if the total length of the selected sentences exceeds 200 words. We set $d_{model} = 512$ for all the embedding vectors. To learn character-level embeddings, we use 512 1D filters for CNN. We set $L = 6, h = 8, d_k = 64, d_v = 64, d_{ff} = 2048$ in Transformer across all our models. The sentence-selector and extractor-generator has about 41.6 and 54.2 million parameters, respectively. We use the output of the 3rd layer ($l = 3$) of the decoder to generate the part-of-speech tags of the keyphrases.

Training. We set $\alpha = 0.7$ and $\beta = 0.5$ for multi-task training. Loss weights for positive samples ω are set to 0.7 and 2.0 (Eq. (4)) during selector and extractor training. We train all our models using Adam [25] with a batch size of 80 and a learning rate of 10^{-4}. During training, we use a dropout rate of 0.2 and a gradient clipping of 1.0. We halve the learning rate when the validation performance drops and stop training if it does not improve for five consecutive iterations. Training the sentence-selector and extractor-generator takes roughly 10 and 25 hours on two GeForce GTX 1080 GPU, respectively.

Testing. The absent keyphrases are generated as a concatenated sequence of words. Hence, unlike prior works [4–6, 35, 58], we use greedy search as the decoding algorithm during testing, and we force the decoder never to output the same trigram more than once to avoid repetitions in the generated keyphrases. This is accomplished by not selecting the word that would create a trigram already exists in the previously decoded sequence. It is a well-known technique utilized in text summarization [38]. Unlike Meng et al. [35], we do not remove single-word predictions for all the testing datasets. We report the averages results of three different random seed.

4.4 Main Results

In this section, we present the experimental results of the baseline methods and our model on present keyphrase extraction and absent keyphrase generation. The results are presented in Table 2 for the scientific domain and Table 3 for the web domain.

Present phrase prediction. From the present keyphrase prediction results, it is evident that SEG-Net outperforms all the baseline methods by a significant margin ($p < 0.05$, t-test) on all the experimental datasets. Unlike the baseline methods, SEG-Net extracts the present keyphrases from the salient sentences, that contributes most to the performance improvement. We observe that SEG-Net performs substantially better than the baselines for the NUS and SemEval datasets. This significant improvement is driven by the higher number of

[3]https://github.com/kenchan0226/keyphrase-generation-rl

Model	KPTimes		In-house	
	F1@M	F1@5	F1@M	F1@5
Present Keyphrase Generation				
catSeq [53]	0.370	0.224	0.255	0.102
catSeqD [53]	0.372	0.233	0.252	0.100
catSeqCorr [4]	0.371	0.230	0.247	0.100
catSeqTG [6]	0.377	0.243	0.260	0.103
catSeq (Transformer)	0.374	0.236	0.258	0.111
SEG-Net (This work)	**0.393**†	**0.341**†	**0.273**†	**0.168**†
Absent Keyphrase Generation				
catSeq [53]	0.119	0.074	0.041	0.020
catSeqD [53]	0.126	0.079	0.037	0.019
catSeqCorr [4]	0.120	0.080	0.037	0.019
catSeqTG [6]	0.129	0.079	0.037	0.018
catSeq (Transformer)	0.128	0.078	0.042	0.020
SEG-Net (This work)	**0.182**†	**0.119**†	**0.048**†	**0.024**†

Table 3: Keyphrase prediction results on the two web domain benchmarks. The bold-faced values indicate the best performances and † means the model is statistically significantly better (by paired bootstrap test, p < 0.05) than all other models.

Model	Present		Absent	
	F1@M	F1@5	F1@M	F1@5
SEG-Net	0.381	0.323	0.038	0.020
w/o sentence selector	0.371†	0.301†	0.037	0.020
w/o layer-wise coverage	0.379	0.321	0.034†	0.016†
w/o "informed" copy	0.381	0.322	0.035†	0.017†
w/o multi-task learning	0.380	0.322	0.037	0.019

Table 4: Ablation study of SEG-Net using the KP20k dataset by precluding the sentence selector, the proposed layer-wise coverage and informed copy attention mechanisms, and the multi-task learning training. † indicates significantly lower (by paired bootstrap test, p < 0.05) performance than SEG-Net.

Input features	Present	
	F1@M	F1@5
Word embedding	0.374	0.317
+ Char-level embedding	0.376	0.318
+ Segment embedding	0.378	0.320
+ Part-of-speech embedding	0.381	0.323

Table 5: Impact of different components of the embedding layer as depicted in Figure 2 on the KP20k dataset.

Dataset	Domain	SEG-Net		Baseline	
		Precision	Recall	Precision	Recall
KP20k	Scientific	84.5	86.3	75.1	95.0
Inspec	Scientific	95.0	82.6	87.1	98.8
Krapivin	Scientific	85.5	85.3	75.8	95.1
NUS	Scientific	91.8	81.0	78.1	92.0
SemEval	Scientific	97.1	75.7	83.5	90.3
KPTimes	Web	81.7	44.9	73.0	45.7
In-house	Web	82.9	49.1	66.8	51.3

Table 6: Results of salient sentence selection on the experiment datasets. The precision and recall are computed by selecting N *predicted* salient sentences in SEG-Net, and the *first* N sentences from the target document in the baseline. We set N for each target document so that the total length of the selected sentences does not exceed a limit of 200 words. It is important to note that the baseline recall is close to 100.0 for the scientific domain datasets because the average length of the target documents from that domain is closer to 200 words.

keyphrases for target documents in those two datasets (see Table 1). We anticipate that SEG-Net is capable of predicting more present keyphrases because it extracts them, in comparison to the baseline methods that generate all the keyphrases.

Absent phrase prediction. Unlike present phrases, absent phrases do not appear exactly in the target document. Hence, predicting them is more challenging and requires a comprehensive understanding of the underlying document semantic. From Table 2 and 3, we see that SEG-Net correctly generates more absent keyphrases than the baselines on all the experimental datasets, except NUS. It is important to note that NUS and SemEval datasets have a different data distribution than KP20k in terms of document lengths and number of keyphrases. During our preliminary experiments, we observed that keyphrase generation models do not generate more absent phrases for NUS and SemEval datasets, comparing to other scientific datasets. Hence, we do not see any substantial performance difference among the experimental models for absent keyphrase prediction on these two datasets. SEG-Net results in the biggest performance improvement (5.4% in terms of F1@M) in the KPTimes dataset and we credit the salient sentence selector for such an improvement (more discussion in section 4.5). Overall, the absent phrase prediction results indicate that SEG-Net is capable of understanding the underlying document semantic better than the baseline methods.

4.5 Ablation Study

We perform a detailed ablation study and the results are reported in Table 4 and 5. We discuss our findings on the following aspects.

Impact of salient sentence selection. One of the key innovations in SEG-Net is the sentence-selector that identifies the salient sentences to minimize the risk of missing critical points due to the truncation of a long target document (e.g., web documents). The contribution of sentence-selector in present keyphrase prediction is significant, as shown in Table 4. Our experiments on the web domain datasets showed that salience sentence selection significantly improved both present and absent keyphrase prediction.

The accuracy of the sentence-selector on the KP20k, KPTimes, and in-house dataset is 78.2, 73.7, and 72.8 in terms of F1 score, respectively. We evaluate SEG-Net on the KP20k dataset given the ground-truth salient sentences to quantify the amount of improvement, achievable with a perfect sentence-selector. Our experiment showed that the performance of SEG-Net for present keyphrase prediction would have increased by approx. 4% in terms of F1@M (from 0.381 to 0.413) with a perfect sentence-selector. To further assess the accuracy of the sentence selector, we compare SEG-Net with a baseline based using precision and recall on predicting the salient sentences from the target documents. The results are presented in

Model	Present		Absent	
	MAE	Avg. #	MAE	Avg. #
Oracle	0.000	2.837	0.000	2.432
catSeq [53]	2.271	3.781	1.943	0.659
catSeqD [53]	2.225	3.694	1.961	0.629
catSeqCorr [4]	2.292	3.790	1.914	0.703
catSeqTG [6]	2.276	3.780	1.956	0.638
SEG-Net (this work)	**2.185**	3.796	**1.324**	1.140

Table 7: Evaluation on predicting the correct number of keyphrases on the KP20k dataset. MAE stands for mean absolute error and "Avg. #" indicates the average number of generated keyphrases per document. Oracle is a model that always generates the ground-truth keyphrases.

Table 6. The higher precision of SEG-Net indicates that it processes input texts with more salient sentences than the baseline. On the other hand, the recall is substantially lower for the scientific domain due to false negative predictions. Overall our experiments suggest that salient sentence selection has a positive impact on keyphrase generation; however, there is further room for improvement.

Impact of coverage and copy attention. We introduce a novel layer-wise coverage attention and an informed copy mechanism in extractor-generator of SEG-Net. From the results shown in Table 4, it is evident that both the attention mechanism helps in improving keyphrase prediction performance. We observed that layer-wise coverage attention results in notable improvements across all datasets. The 'informed" copy mechanism does not allow to copy a present phrase term during absent keyphrase generation. Hence, we speculate the impact of the "informed" copy mechanism would depend on the overlapping amount among the present and absent keyphrases. For example, in the KP20k and KPTimes test dataset, 7.5% and 3.8% of absent phrase terms overlap with present phrase terms. Our experimental findings corroborate with our speculation as we observed significant improvements in KP20k, while no substantial gain for the KPTimes dataset because of the informed copy mechanism.

Impact of multi-task learning. To provide syntactic supervision, we train the extractor-generator on keyphrase and their part-of-speech (POS) tag generation via multi-task learning (MTL). From Table 4, we see there is no significant drop in performance when we do not apply MTL for the KP20k dataset. However, we observed MTL substantially impacts absent keyphrase generation on the KPTimes dataset. It is important to note that our goal behind applying MTL is to reduce generating overlapping keyphrases. Hence, we evaluated the generated absent keyphrases by checking if they are super-phrase or sub-phrase of another phrase. We found that due to MTL, SEG-Net generates less number of duplicate and overlapping phrases, and improves in accurate number of keyphrase generation.

Impact of embeddings. In the embedding layer of both sentence-selector and extractor-generator, we learn five different embedding vectors: word embedding, character-level embedding, position embedding, segment embedding, and part-of-speech (POS) embedding that are element-wise added. With word embedding and position embedding remained intact, we successively added the other three embedding vectors and observed performance improvements on present keyphrase prediction task. The results are presented in Table 5. We see the most significant gain comes from the usage of POS embedding as it helps SEG-Net to distinguish between phrase patterns. The addition of segment embedding is also useful, specially since the sentence-selector may predict salient sentences from any part of the document, and we hypothesize that the sentence index guides the self-attention mechanism in the extractor-generator. The character-level embeddings are employed as we limit the vocabulary to the most frequent V words. We observed that character-level embeddings have a notable impact in the web domain, where the actual vocabulary can be enormous. We performed the same study on sentence-selector and observed a similar trend in the performance.

Model	Present		Absent	
	F1@M	F1@5	F1@M	F1@5
catSeq [53]	0.376	0.298	0.034	0.016
catSeqD [4]	0.372	0.293	0.033	0.016
catSeqCorr [6]	0.375	0.300	0.034	0.016
catSeqTG [3]	0.374	0.302	0.033	0.016
SEG-Net (this work)	**0.390**	**0.326**	**0.042**	**0.021**

Table 8: Keyphrase prediction results on the KP20k dataset with "name variations" as proposed in Chan et al. [3].

4.6 Analysis

We analyze SEG-Net's performance on generating (1) an accurate number of keyphrases and (2) variation of named entities.

Number of generated keyphrases. Generating an accurate number of keyphrases indicates a model's understanding of the documents' semantic. For example, a small number of phrase predictions demonstrate a model's inability to identifying all the key points expressed in a document. On the other hand, over generation implies a model's wrong understanding of the crucial points. Hence, we compare our model with all the baseline methods on predicting the appropriate number of phrases. Following Chan et al. [3], we measure the mean absolute error (MAE) between the number of generated keyphrases and the number of ground-truth keyphrases. We remove duplicated keyphrases before evaluation. The results are presented in Table 7. The lower MAEs for SEG-Net demonstrate that our model notably outperforms the baselines on predicting the number of absent keyphrases and slightly outperforms the baselines on predicting the number of present keyphrases. We want to emphasize that the notable reduction in MAE for absent phrase generation is due to the joint training of SEG-Net with part-of-speech (POS) tagging. We anticipate that with two generative tasks at hand, the model learns to gauge the number of keyphrases for generation accurately.

Generating variation of named entities. A keyphrase can be expressed in different ways, such as, "solid state drive" as "ssd" or "electronic commerce" as "e commerce" etc. Hence, Chan et al. [3] aggregated name variations of the ground-truth keyphrases from the KP20k evaluation dataset using the Wikipedia knowledge base. We evaluate our model on that enriched evaluation set, and the experimental results are listed in Table 8. We observed that although SEG-Net extracts the present keyphrases, it is capable of predicting present phrases with variations such as "support vector machine" and "svm" if they co-exist in the target document.

5 CONCLUSION

In this paper, we present SEG-Net, a keyphrase generation model that identifies the salient sentences in a target document to utilize maximal information for keyphrase prediction. SEG-Net jointly learns to predict both present and absent keyphrases from the target document. In SEG-Net, we incorporate novel layer-wise coverage and an informed copy attention to cover all the critical points in a document and diversify the present and absent keyphrases. We jointly train SEG-Net for keyphrase generation and their part-of-speech tags prediction. We evaluate SEG-Net on five benchmarks from scientific articles and two benchmarks from web documents. The experiment results demonstrate that it is effective over the state-of-the-art neural generative methods on both scientific and web domains.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] Gábor Berend. 2011. Opinion Expression Mining by Exploiting Keyphrase Extraction. In *Proceedings of 5th IJCNLP*. 1162–1170.

[3] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *Proceedings of the 57th ACL*. 2163–2174.

[4] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase Generation with Correlation Constraints. In *Proc. of the EMNLP*. 4057–4066.

[5] Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019. An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction. In *Proceedings of the 2019 NAACL*. 2846–2856.

[6] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proc. of the AAAI*. 6268–6275.

[7] Yen-Chun Chen and Mohit Bansal. 2018. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In *Proceedings of the 56th ACL*. 675–686.

[8] Kushal S Dave and Vasudeva Varma. 2010. Pattern based keyword extraction for contextual advertising. In *Proceedings of the CIKM*. ACM, 1885–1888.

[9] Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. Topicrnn: A recurrent neural network with long-range semantic dependency. *ICLR*.

[10] Ygor Gallina, Florian Boudin, and Beatrice Daille. 2019. KPTimes: A Large-Scale Dataset for Keyphrase Generation on News Documents. (2019), 130–135.

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

[12] Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of the 31st AAAI*.

[13] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the 30th ICML*. 1319–1327.

[14] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of WWW*. 661–670.

[15] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*. 1631–1640.

[16] Khaled M Hammouda, Diego N Matute, and Mohamed S Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *MLDM*. 265–274.

[17] Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*. 7944–7954.

[18] Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the EMNLP*. 216–223.

[19] Anette Hulth and Beáta B Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of COLING-ACL*. 537–544.

[20] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. 448–456.

[21] Steve Jones and Mark S Staveley. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the SIGIR*. ACM, 160–167.

[22] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th SemEval*. 21–26.

[23] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the EMNLP*. 1746–1751.

[24] Youngsam Kim, Munhyong Kim, Andrew Cattle, Julia Otmakhova, Suzi Park, and Hyopil Shin. 2013. Applying graph-based keyword extraction to document retrieval. In *Proceedings of the Sixth IJCAI*. 864–868.

[25] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[26] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. *Large dataset for keyphrases extraction*. Technical Report. University of Trento.

[27] Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 665–671.

[28] Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring Sentence Singletons and Pairs for Abstractive Summarization. In *Proceedings of the 57th ACL*. 2175–2189.

[29] Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proc. of CoNLL*. 135–144.

[30] Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th SemEval*. 248–251.

[31] Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific Information Extraction with Semi-supervised Neural Tagging. In *Proc. of EMNLP*. 2641–2651.

[32] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*. 1412–1421.

[33] Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the EMNLP*. 1318–1327.

[34] Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with Wikipedia. In *Proceedings of the AAAI WikiAI workshop*, Vol. 1. 19–24.

[35] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. (2017), 582–592.

[36] Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and Robust Question Answering from Minimal Context over Documents. In *Proceedings of the 56th ACL*. 1725–1735.

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

[38] Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

[39] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*. 1073–1083.

[40] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 NAACL*. 464–468.

[41] Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural Models for Key Phrase Extraction and Question Generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*. 78–88.

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). 3104–3112.

[43] Avinash Swaminathan, Raj Kuwar Gupta, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, and Rajiv Ratn Shah. 2019. Keyphrase Generation for Scientific Articles using GANs. *arXiv preprint arXiv:1909.12229* (2019).

[44] Yixuan Tang, Weilong Huang, Qi Liu, Anthony KH Tung, Xiaoli Wang, Jisong Yang, and Beibei Zhang. 2017. Qalink: enriching text documents with relevant Q&A site contents. In *Proceedings of the CIKM*. ACM, 1359–1368.

[45] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of ACL*. 76–85.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[47] Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge.. In *AAAI*, Vol. 8. 855–860.

[48] Minmei Wang, Bo Zhao, and Yihua Huang. 2016. Ptr: Phrase-based topical ranking for automatic keyphrase extraction in scientific publications. In *International Conference on Neural Information Processing*. Springer, 120–128.

[49] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the EMNLP*.

[50] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*. 129–152.

[51] Xiaoyuan Wu and Alvaro Bolivar. 2008. Keyword extraction for contextual advertisement. In *Proceedings of the WWW*. ACM, 1195–1196.

[52] Hai Ye and Lu Wang. 2018. Semi-Supervised Learning for Neural Keyphrase Generation. In *Proceedings of the EMNLP*. 4142–4153.

[53] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2018. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. *arXiv:1810.05241* (2018).

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

[55] Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. Mike: keyphrase extraction by integrating multidimensional information. In *Proceedings of the CIKM*. ACM, 1349–1358.

[56] Yong Zhang, Yang Fang, and Xiao Weidong. 2017. Deep keyphrase generation with a convolutional sequence to sequence model. In *ICSAI*. IEEE, 1477–1485.

[57] Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelli. and Agent Sys.* 2, 1 (2004), 39–53.

[58] Jing Zhao and Yuxiang Zhang. 2019. Incorporating Linguistic Constraints into Keyphrase Generation. In *Proceedings of the 57th ACL*. 5224–5233.