TransKP: Transformer based Key-Phrase Extraction

Mukund Rungta*, Rishabh Kumar*, Mehak Preet Dhaliwal*, Hemant Tiwari , Vanraj Vala
Samsung R&D Institute Bangalore, Karnataka, India 560037
Email: {mukund.r, rish.kumar, m.dhaliwal, h.tiwari, vanraj.vala}@samsung.com

***Abstract*—Increased connectivity has led to a sharp rise in the creation and availability of structured and unstructured text content, with millions of new documents being generated every minute. Key-phrase extraction is the process of finding the most important words and phrases which best capture the overall meaning and topics of a text document. Common techniques follow supervised or unsupervised methods for extractive or abstractive key-phrase extraction, but struggle to perform well and generalize to different datasets. In this paper, we follow a supervised, extractive approach and model the key-phrase extraction problem as a sequence labeling task. We utilize the power of transformers on sequential tasks and explore the effect of initializing the embedding layer of the model with pre-trained weights. We test our model on different standard key-phrase extraction datasets and our results significantly outperform all baselines as well as state-of-the-art scores on all the datasets.**

***Index Terms*—keyphrase extraction, transformer, seq2seq**

I. Introduction

An explosive growth in the generation and consumption of textual data has rendered the issue of analysing, comprehending and processing text documents inevitable. With over millions of emails, social media posts, blogs, articles, messages etc. being shared every day, extracting the most important information which best captures the meaning of the text is important for both the humans reading that document, as well as for downstream text processing tasks. Key-phrase extraction refers to the process of finding the most important words and phrases that accurately provide an overview of the document. Automated key phrase extraction becomes critical at the current document creation rate, enabling document tagging, summarization, topic analysis, document clustering and classification, as well as various information retrieval tasks such as document indexing and improving search engine results. All this makes the process of key-phrase extraction from text a crucial task in the field of Natural Language Processing (NLP).

A popular survey on key-phrase extraction [1] shows that there are two main approaches for the task. The first approach involves candidate key-phrase generation from the document, followed by their pruning and ranking. Extraction of the candidates employs different heuristics such as named entity extraction, position of the phrases and their frequency of occurrence in the document. This method suffers from the limitation of not being able to generalize to different document types due to the application of domain specific heuristics. Also, the candidate generation method does not consider the dependency between different key-phrases. In order to overcome the above limitations, [2] modelled the problem as a sequence labeling task where each token in the document is assigned a tag, indicating whether it is a part of key-phrase or not. This method captures the long term dependencies in the document to detect its important key-phrases.

The learning method for key-phrase extraction can be supervised or unsupervised. Unsupervised approaches do not require a labeled dataset with already tagged key-phrases, but instead exploit the underlying structure of the text documents, typically using statistical [3] or graph based techniques [4], [5]. Supervised approaches generally employ NLP machine learning or deep learning techniques on the labeled dataset [6]. Depending on the kind of key-phrases extracted, the different methodologies can also be categorised as extractive or abstractive, where extractive approaches [4], [7] select the most important words and phrases from the document, whereas abstractive approaches [8] can additionally generate key-phrases which were not part of the original document.

While the task of key-phrase extraction has been explored, results on standard evaluation metrics have not been very promising. Some of the key issues identified in [1] relate to the kind of text corpora available, with complexity increasing with the length of the documents, structural inconsistencies between different kinds of documents, changes in topics within a document and possibly uncorrelated different topics present in the same document. We, therefore, propose an extractive, supervised approach to extract the key-phrases of a document by modelling it as a sequence labeling task. In this model, we leverage the power of the transformer [9], which has previously showcased its ability to capture long term dependencies and relations between different components of the document using positional encoding and multi-head self-attention. We name our transformer based key-phrase extraction model 'TransKP'. Additionally, we explore the effect of initialization of the embedding layer of the transformer and empirically prove that this significantly improves results on our task. To our knowledge, we are the first to employ transformers to achieve the objective of key-phrase extraction as well as studying the effect of initializing transformer layers on the task. Our results on all the datasets prove the learning capability as well as generalisation power of our network.

We have evaluated our model against various baselines as well as state-of-the-art models on the key-phrase extraction task and show significantly improved results on the evaluation metrics on multiple standard datasets, thus proving the effectiveness of our experiments and model. The key-phrases extracted by our model along with the gold standard labels on

* These authors contributed equally to the paper

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

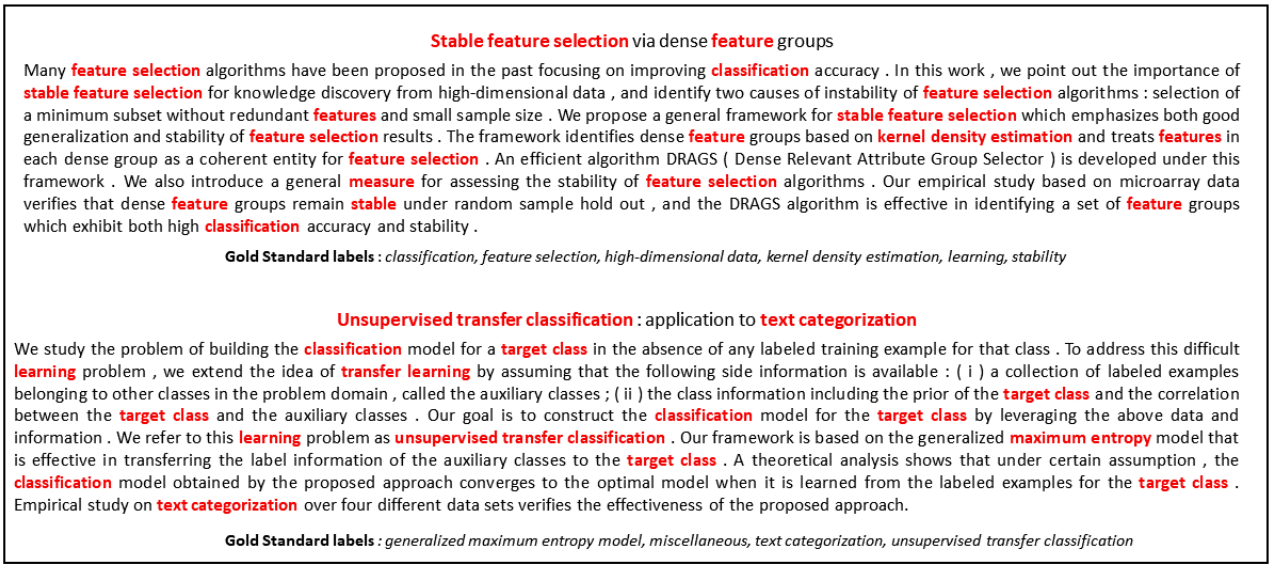

Fig. 1. Key-phrases (In bold) detected by TransKP on two sample documents

two sample documents from our testing set can be visualised in Fig. 1. Summarizing our contributions:

- We explore the transformer based model for the task of key-phrase extraction by formulating it as a sequence labeling problem.
- We compared the results of our model- TransKP against the standard baselines and state-of-the-art model on multiple standard datasets and report significant improvement over them.
- We investigate the effect of initializing the model with pre-trained embedding weights (Trans-KP Glove) against random initialization (Trans-KP). The results indicate superior performance, both in terms of F1-score and convergence time, of Trans-KP Glove with respect to Trans-KP.

II. Related Work

Most key-phrase extraction techniques follow a two-step approach, with the first step being identification of candidate key-phrases followed by a final predication of each candidate as a key-phrase (KP) or non-key-phrase (non-KP). The second phase can follow a supervised or unsupervised approach.

Unsupervised approaches typically follow graph-based extraction techniques. These methods involve transformation of the input document to a graph structure with each candidate phrase represented as a node and edges between the nodes indicating some kind of syntactic or semantic similarity. A final score is calculated for each node following a graph-based ranking method which determines the corresponding candidate's relevance as a key-phrase for the document. Several methods utilise variants of the popular PageRank algorithm [10] which is used to rank web-pages based on their incoming and outgoing links. One of the most popular key-phrase extraction algorithms which follows this approach is the TextRank algorithm [4]. An extension of this method is the SingleRank algorithm [5] which exploits word co-occurrence information to determine edge-weights. Other approaches such as PositionRank [11] and TopicalPageRank [12], additionally utilise features such as word position and topical information with the traditional TextRank algorithm. Some recent approaches [13], [14] have also incorporated word embeddings in unsupervised key-phrase extraction. Various other approaches such as TopicRank [15], KeyCluster [16] and CommunityCluster [17] perform topic based clustering to ensure sufficient relevance and coverage of the extracted key-phrases with respect to the topics of the document [1].

Supervised approaches for key-phrase extraction typically model the problem as binary classification of the tokens present in the document into KP or non-KP. Different learning algorithms like naïve Bayes [18] decision tree [19], SVM [20], boosting [21], bagging [22] have been used to train classifiers. Several classifiers are also trained on features based on document properties like statistical properties [18] [23] , location based selection [24] and syntactic properties [25]. A few works have also focused on extracting information from external resources such as [26] have used Wikipedia information and [27] have used citation resources to enhance key-phrase extraction from documents.

With new advancements in the field of deep learning, several neural models have been proposed to extract key-phrases from documents. Authors of [28] have proposed the method of using joint layer Recurrent Neural Network (RNN) to capture semantic dependencies in the input sequence for extracting

key-phrases from short documents like Twitter posts. However, this model suffers from the limitation of low performance on larger documents. The authors of [8] formulated the problem as key-phrase generation rather than extraction. They propose an Encoder-Decoder model with copying mechanism to generate key-phrases from the input document. Since we have formulated our problem as sequence labeling task, we focus on extracting key-phrases present in the document unlike [8], which follows an abstractive approach. Recently, [2] have formulated key-phrase extraction as sequence labeling task by training a Conditional Random Field (CRF) using linguistic, surfaceform, and document-structure information for predicting the label for each token in the document as KP or non-KP. Authors of [7] have incorporated Bi-LSTM with Conditional Random Field for capturing the hidden semantics in the text through long distance dependencies. Since, this model has claimed to outperform the existing baselines and previous approached we have used it as one of our baseline for comparison.

Our approach follows the aforementioned works in formulating the task as a sequence labeling problem while extending the labeling scheme to incorporate inter-word dependencies between phrases. In our knowledge, we are the first to utilise the power of the transformer to capture long-term dependencies between tokens in a document to extract relevant key-phrases.

III. Methodology

In this section, we formally describe our formulation of the key-phrase extraction task as a sequence labeling problem. We also explain our encoder-decoder architecture which is based on [9]. We further analyse the need of using pre-trained word embedding for initializing the encoder weights over random values.

A. Problem Formulation

Given a document D, represented as a sequence of words $D = [W_1, W_2, \dots W_n]$ where W_i represents the i_{th} word in the document, our task is to assign each word a label:

- **B** if it is the beginning of a key-phrase
- **I** if it is a subsequent part of the key-phrase
- **O** if it is not a part of any key-phrase

This labeling scheme follows the work of [29]. Modelling key-phrase extraction as a sequence to sequence task enables us to exploit the relationships between different words and their combined role in representing the document. An alternative labeling scheme is to assign a binary label to words in the document representing whether it is part of a key-phrase or not [7]. However, this approach fails to capture the inter-word relationships within a phrase.

B. Encoder - Decoder

The Encoder-Decoder architecture has proven its usefulness in a multitude of sequence to sequence tasks [30]–[32]. It involves using one part of the network to encode the inputs to vectors in a latent space and using the other part to decode these vectors in the output space. It enables us to train a single model, end to end, directly on source-target pairs while also handling variable sized inputs.

The transformer model, first introduced in [9] captures the sequential nature of data with the help of multi-head attention units. It consists of 6 stacked layers of encoder and decoder modules. Each encoder module consists of a multi-headed self-attention unit followed by a feed forward network. The decoder modules follow the architecture of the encoder but additionally incorporate a multi-headed attention unit over the outputs of the encoder stack. The complete architecture of the transformer can be visualised in Fig. 2.

C. Positional Encoding

An important part of any sequence to sequence architecture is the knowledge of the position of each word in the input and target sequence. This information is encoded by creating position specific values as mentioned in the equation below

$$\begin{aligned} PE_{(pos,2i)} &= sin(\frac{pos}{10000^{2i/d_{model}}}) \\ PE_{(pos,2i+1)} &= cos(\frac{pos}{10000^{2i/d_{model}}}) \end{aligned} \tag{1}$$

where 'pos' refers to the order of the word in the document, 'i' refers to its position in the embedding vector and 'd_{model}' is the dimension of the embedding layer. These equations generate a constant 2D matrix which is added to the embeddings of the words, thus incorporating positional information in them. However, to ensure that original information in embedding vectors is not lost, we make positional encodings relatively smaller than the embedding vector's values.

D. Multiheaded Self Attention

The attention mechanism, first introduced in [33] is a method of capturing the importance of words in a sequence with respect to other words. When attention values are calculated among words within the same sequence, we refer to it as self-attention.

To obtain these values, each input embedding token, position encoded and masked, is projected into three vectors called the 'Key', 'Query' and 'Value' vectors through trainable weight matrices. To get the attention score of a word with respect to the other words in the sequence, we take the dot product of its 'Query' vector with the 'Key' vector of all the other words and update its 'Value' vector with respect to each word. In matrix form, the equation can be seen in the equation below.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}).V \tag{2}$$

A high softmax score indicates that the two words are relevant to each other and thus the corresponding 'Value' vector is high. For multi-headed attention, the 'Key', 'Query' and 'Value' vectors are split into 'n' heads with each head learning from a different context at the same time. Thus, we get multiple sets of these 'Values' for each word from the different contexts

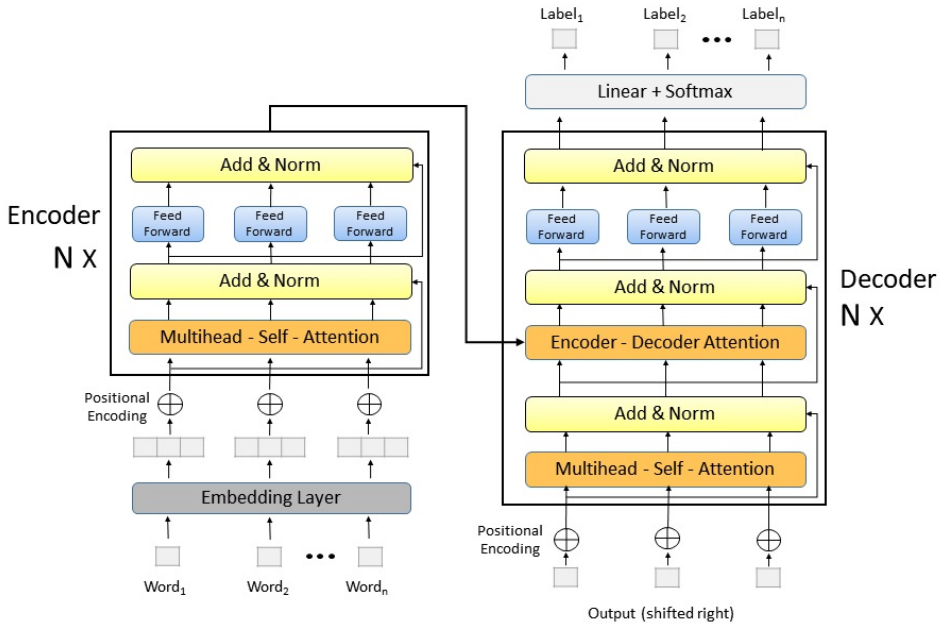

Fig. 2. The Transformer Model Architecture

which are then concatenated and multiplied with a final learned matrix to reduce the multiple embedding sets to a single embedding. Hence, the attention layer converts position aware naive form of embeddings to a more context aware embedding. In the decoding stack this multi headed attention is calculated first with the outputs predicted till that time step and, after normalization, with all the encoded words from the input sequence. This establishes the relevance of current output with both the outputs predicted till now as well as neighbouring words in the input.

E. Normalization and Feed Forward Neural Network

Layer normalization [34] is performed after each unit to ensure that the range of values doesn't change too much and the model converges faster. It normalizes the input across the features unlike batch normalization, which normalizes each feature across the batch. This has experimentally been seen to perform better. This normalization also adds the inputs from the previous layers as residual connections, thus allowing the model to retain previous information.

Each block further contains a feed forward neural network, consisting of two linear layers with 'ReLU' activation. The input to this network is the normalized embedding with attention, which is mapped to a new embedding in a latent space common to the whole language.

As this unit of the network does not have any interdependencies between different parts of the input, it enables parallel computation thus reducing the training time and resources required.

F. Weighted Cross Entropy Loss

For key phrase extraction task, the 'BIO' labels will be highly unbalanced since key phrases constitute a very small portion of input text. To give a higher weightage for correct prediction of 'B' and 'I' labels, we calculate the weights of each label according to their relative occurrence in training data for calculating the cross entropy loss. This results in label 'I' getting the highest weight and the label 'O' the least.

G. Initialization of Embedding Layer

The transformer starts with random initialization for embedding weights and gradually trains them to be more context aware. However, initializing the embedding layer with pre trained weights like Glove [35] which already has some contextual information gives two fold advantages:

- It allows the model to use external knowledge and then fine tune it according to inter word relationships
- It results in faster convergence, thus, reducing the training time.

This initialization helped us to achieve much better results more quickly as explained in Section IV.

IV. Experiments

In this section we describe the details of the datasets used for training and evaluation of the proposed model. In addition to this, we discuss the implementation details of the proposed model, hyper parameters, baselines and metrics used for comparison.

A. Dataset

We used three publicly available datasets for evaluation of the model. For the purpose of training, we have used the dataset provided by [8] which consists of high-quality scientific metadata in the computer science domain from online digital libraries like ACM Digital Library, ScienceDirect, Wiley and Web of Science. This dataset contains 567,830 papers with pre-defined training, validation and testing split provided by the authors as follows: 527,830 for training (kp527k), 20,000 for validation (kp20k-v) and the remaining 20,000 for testing (kp20k). Evaluation of the proposed model is also performed on publicly available KDD and WWW datasets [2].

- **KDD**– This dataset is provided by [2]. It contains abstracts and author annotated key-phrases for 755 research papers published in ACM Conference on Knowledge Discovery and Data mining (KDD). The entire dataset is used for testing the performance of the proposed model.
- **WWW**– This dataset is also provided by [2] and contains abstracts and author annotated key-phrases for 1330 research papers published in World Wide Web Conference (WWW). This entire dataset is used for testing the performance of the proposed model as well.

All the mentioned datasets contain the title, abstract and author-labeled key phrases. The title and abstract of each paper are combined together to extract key-phrases and author-labeled key-phrases are used as gold standards for evaluation. We have trained the model using the kp527k dataset and validated on the kp20k-v dataset. This trained model is used for evaluation of the performance on all the three testing datasets kp20k, KDD and WWW. Statistics of the training and testing dataset are mentioned in the Table I.

TABLE I
DETAILS OF DATASETS

Type	Dataset	Number of docs	Avg Key-Phrases per doc
Training	kp527k	527,830	5.3
Validation	kp20k-v	20,000	5.3
Testing	kp20k	20,000	5.3
	KDD	755	4.1
	WWW	1,330	4.8

B. Implementation Details

As mentioned in the previous section, we have formulated the key-phrase extraction problem as a sequence labeling task where each token of title and abstract is mapped to one of $\{B, I, O\}$, with 'B' indicating the start of key-phrase, 'I' indicating presence inside a key-phrase and 'O' indicating that it is not a part of any key-phrase. Following an analysis of the training data, we have only considered the documents having a maximum length of 200 words for training purpose.

Since the labeled data is imbalanced towards the 'O' category as compared to 'B' and 'I', we give a higher weightage to the latter by normalizing their frequency of occurrence in the training data. These values serve as the weights for the Categorical Cross Entropy for loss calculation, the weight being 0.78 for 'B', 1.0 for 'I' and 0.13 for 'O'.

As mentioned in the previous section, the embedding layer of the encoder is initialized in two different manners, one being random initialization (TransKP) and other being 300 dimension Glove pre-trained embedding vectors (TransKP-Glove). The dimension of the hidden layer is set to 300 in both the cases.

C. Baselines and Metrics

Comparison of the proposed method is performed against several state-of-the-art models and baselines – Bi-LSTM-CRF [7], copy-RNN [8], KEA [18], Tf-Idf, TextRank [4] and SingleRank [5]. We used the data for baseline results from [7]. The evaluation procedure followed is consistent with previous works and baseline measures, in which model predicted keyphrases are evaluated against gold-standard keyphrases and the Precision, Recall and F1-score are calculated. The comparison is focused on the F1-score which is the harmonic mean of the Precision and Recall.

V. RESULTS

This section shows the evaluation of the proposed model for the key-phrase extraction problem and its comparison against the baselines mentioned in the previous section.

A. Performance of TransKP

Evaluation of proposed TransKP model is performed on kp20k, WWW and KDD datasets. For any document, output of the TransKP model is a sequence of labels corresponding to the words in the document, indicating whether they are part of a key-phrase or not. The candidate key-phrases are constructed based on these labels and are further used for calculating the performance of the model.

From Table II, we notice that for the kp20k dataset F1-score is 44.12% using the TransKP-Glove model and 36.71% for the randomly initialized TransKP model. The TransKP-Glove model also shows an increase in both the precision and recall values. These results indicate that the Glove initialized model captures an efficient representation of the document which is used for predicting the key-phrases. A similar trend is seen for WWW and KDD dataset as well. The F1-score for the TransKP-Glove model on WWW dataset is 48.22% and for the randomly initialized TransKP it is 41.34%. On the KDD dataset, F1-score for TransKP-Glove is 41.56% whereas it is 39.77% for randomly initialized TransKP.

Fig. 1 shows examples of key-phrase extraction on the abstract of two scientific papers from KDD dataset. Key-phrases marked bold are the predictions from the proposed TransKP-Glove model and the gold standard author annotated key-phrases are mentioned below the abstract. From the figure it can be inferred that our model is successful in predicting the important key-phrases. It is able to predict multi-word

TABLE II
COMPARISON OF TRANSKP WITH BASELINES ON KP20K, WWW AND KDD DATASETS

Method	kp20k			WWW			KDD		
	Precision	*Recall*	*F1-Score*	*Precision*	*Recall*	*F1-Score*	*Precision*	*Recall*	*F1-Score*
TransKP-Glove	40.48	48.49	**44.12**	40.46	59.67	**48.22**	32.62	57.27	**41.56**
TransKP	33.55	40.51	36.70	35.73	50.31	41.78	33.32	49.32	39.77
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	11.80	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	13.83	11.39	14.50	12.42

key-phrases like "unsupervised transfer classification" and "stable feature selection". It can also predict derivatives of gold standard key-phrases. For example, the gold standard has two key-phrases "stability" and "feature selection", however, our model predicted "stable feature selection" as a key-phrase which conveys a better meaning than the original two words. Although, this improves the real world performance, it has an adverse effect on our F1-score.

We have also compared the training loss for the TransKP-Glove model against TransKP. Fig. 3 shows the loss incurred during training of both the models. From the plot it is evident that the loss values for the model initialized with Glove embedding reduced faster than random initialization. This result is consistent with the literature [36], where it is stated that training a model with rich linguistic pretrained embedding converges quickly in comparison to a model with randomly initialized embedding layer.

B. Comparison with Baselines

We compare our TransKP model with several strong baselines and existing state of the art models on the mentioned datasets. The baselines includes a combination of supervised and unsupervised models. The unsupervised baselines are Tf-Idf, TextRank and SingleRank whereas the supervised models are Bi-LSTM-CRF, copyRNN and KEA. The Bi-LSTM-CRF model is the best performing model out of all the baselines.

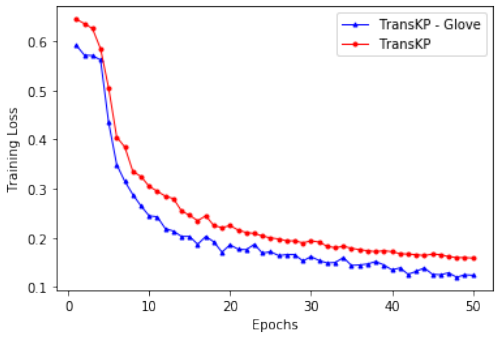

Fig. 3. Training loss curves of TransKP-Glove and TransKP

All the supervised models have been trained on the kp527k dataset.

Table II shows the result of this comparison. Both our transformer based models outperform all the baselines in terms of F1 score on kp20k and WWW dataset. The TransKP-Glove model outperforms all baselines on all three datasets. The F1 scores for kp20k, WWW and KDD datasets show a significant improvement of 8.49%, 8.79% and 0.49% respectively. We also outperformed all other unsupervised learning and deep learning based models including sequence to sequence model, copyRNN, with a significant improvement in precision, recall and F1 scores. Notably, our model shows a huge improvement in recall scores compared to the previous best models. For kp20k the recall is improved from 24.66% to 48.49%, for WWW it shows an increase of 31.24% whereas the increase for KDD is 25.42%. The Bi-LSTM-CRF model has a significantly higher precision value than the recall which, although improves F1 score but can fail to cover all the concepts in the given text. In contrast, our model maintains a balance between precision and recall, without getting biased towards any one parameter, therefore resulting in a better real-world performance.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a transformer based neural model for the task of key-phrase extraction from text documents. We propose an extractive, supervised approach by modelling the task as a sequence to sequence labeling problem and leveraging the power of transformers on sequential tasks. We also explore the effect of initializing the embedding layer of the transformer with pre-trained embedding and empirically prove that such an initialization results in a significant boost of performance. In our knowledge, we are the first to utilize transformers as well as experiment with transformer layer initializations for the task of key-phrase extraction. Our results evaluated on standard datasets and metrics beat the score of all baselines and state-of-the-art models. Significantly improved performance on different datasets proves our model's learning as well as generalization capability.

Future scope includes learning how to rank the different extracted key phrases in terms of their importance in captur-

ing the article's main topics as well as ensuring maximum coverage across all topics. In addition to this, the trained keyphrase architecture can be utilized for the downstream task of text summarization via transfer learning.

REFERENCES

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

[2] S. D. Gollapalli and C. Caragea, "Extracting keyphrases from research papers using citation networks," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[3] M.-S. Paukkeri and T. Honkela, "Likey: Unsupervised language-independent keyphrase extraction," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, ser. SemEval '10. USA: Association for Computational Linguistics, 2010, p. 162–165.

[4] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[5] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge." in *AAAI*, vol. 8, 2008, pp. 855–860.

[6] F. Bulgarov and C. Caragea, "A comparison of supervised keyphrase extraction models," in *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*. ACM Press, 2015. [Online]. Available: https://doi.org/10.1145/2740908.2742776

[7] R. Alzaidy, C. Caragea, and C. L. Giles, "Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents," in *The world wide web conference*, 2019, pp. 2551–2557.

[8] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," *arXiv preprint arXiv:1704.06879*, 2017.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[11] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1105–1115.

[12] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, Oct. 2010, pp. 366–376. [Online]. Available: https://www.aclweb.org/anthology/D10-1036

[13] D. Mahata, J. Kuriakose, R. Shah, and R. Zimmermann, "Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 634–639.

[14] R. Wang, W. Liu, and C. McDonald, "Using word embeddings to enhance keyword identification for scientific publications," in *Databases Theory and Applications*, M. A. Sharaf, M. A. Cheema, and J. Qi, Eds. Cham: Springer International Publishing, 2015, pp. 257–268.

[15] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," 2013.

[16] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, Aug. 2009, pp. 257–266. [Online]. Available: https://www.aclweb.org/anthology/D09-1027

[17] M. Grineva, M. Grinev, and D. Lizorkin, "Extracting key terms from noisy and multitheme documents," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 661–670. [Online]. Available: https://doi.org/10.1145/1526709.1526798

[18] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automated keyphrase extraction," in *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*. IGI global, 2005, pp. 129–152.

[19] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[20] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li, "Topical keyphrase extraction from twitter," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 379–388.

[21] A. Hulth, J. Karlgren, A. Jonsson, H. Boström, and L. Asker, "Automatic keyword extraction using domain knowledge," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2001, pp. 472–482.

[22] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, 2003, pp. 216–223.

[23] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning, "Domain-specific keyphrase extraction."

[24] S. N. Kim and M.-Y. Kan, "Re-examining automatic keyphrase extraction approaches in scientific articles," in *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications*. Association for Computational Linguistics, 2009, pp. 9–16.

[25] T. D. Nguyen and M.-Y. Kan, "Keyphrase extraction in scientific publications," in *International conference on Asian digital libraries*. Springer, 2007, pp. 317–326.

[26] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, 2009, pp. 1318–1327.

[27] C. Caragea, F. A. Bulgarov, A. Godea, and S. D. Gollapalli, "Citation-enhanced keyphrase extraction from research papers: A supervised approach," 2014.

[28] Q. Zhang, Y. Wang, Y. Gong, and X.-J. Huang, "Keyphrase extraction using deep recurrent neural networks on twitter," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 836–845.

[29] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[30] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *Interspeech 2017*. ISCA, Aug. 2017. [Online]. Available: https://doi.org/10.21437/interspeech.2017-1705

[31] R. Nallapati, B. Xiang, and B. Zhou, "Sequence-to-sequence rnns for text summarization," *CoRR*, vol. abs/1602.06023, 2016. [Online]. Available: http://arxiv.org/abs/1602.06023

[32] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," 2015.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[34] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

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

[36] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, 2017.