Expert Systems With Applications 202 (2022) 117096

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

PSRMTE: Paper submission recommendation using mixtures of transformer

Dac Huu Nguyen [a], Son Thanh Huynh [c,d,e], Cuong Viet Dinh [b], Phong Tan Huynh [c], Binh Thanh Nguyen [c,d,e,*]

[a] John Von Neumann Institute, Ho Chi Minh City, Viet Nam
[b] Dublin City University, Ireland
[c] AISIA Research Lab, Ho Chi Minh City, Viet Nam
[d] Department of Computer Science, University of Science, Ho Chi Minh City, Viet Nam
[e] Vietnam National University, Ho Chi Minh City, Viet Nam

ARTICLE INFO

Keywords:
Recommendation system
Deep learning
Transformer encoders

ABSTRACT

Nowadays, there has been a rapidly increasing number of scientific submissions in multiple research domains. A large number of journals have various acceptance rates, impact factors, and rankings in different publishers. It becomes time-consuming for many researchers to select the most suitable journal to submit their work with the highest acceptance rate. A paper submission recommendation system is more critical for the research community and publishers as it gives scientists another support to complete their submission conveniently. This paper investigates the submission recommendation system for two main research topics: computer science and applied mathematics. Unlike the previous works (Wang et al., 2018; Son et al., 2020) that extract TF–IDF and statistical features as well as utilize numerous machine learning algorithms (logistics regression and multiple perceptrons) for building the recommendation engine, we present an efficient paper submission recommendation algorithm by using different bidirectional transformer encoders and the Mixture of Transformer Encoders technique. We compare the performance between our methodology and other approaches by one dataset from Wang et al. (2018) with 14012 papers in computer science and another dataset collected by us with 223,782 articles in 178 Springer applied mathematics journals in terms of top K accuracy ($K = 1, 3, 5, 10$). The experimental results show that our proposed method extensively outperforms other state-of-the-art techniques with a significant margin in all top K accuracy for both two datasets. We publish all datasets collected and our implementation codes for further references.[1]

1. Introduction

Nowadays, there have been many scientific journals in diverse publishers, such as IEEE, ACM, Springer, Elsevier, Science, Oxford University Press, and MIT Press, for each research domain. Each scientific journal has specific scopes, indexes, journal metrics (Impact Factor, CiteScore, SCImago Journal Ranking), and other information. With the massive number of paper submissions into different journals (especially for the top-rank journals or conferences), the competency rate becomes bigger and bigger to get a scientific paper accepted. Consequently, picking an appropriate scientific journal for submission and getting more acceptance opportunities are essential for many researchers. Numerous publishers (e.g., IEEE,[2] Springer,[3] and Elsevier[4]) create their submission recommendation tools to help scientists quickly discover top relevant journals in their systems with a convenient user interface (UI). There are other domain-specific systems that focus on a specific field of research, including PubMender[5] (Feng et al., 2019) and PRS[6] (Wang et al., 2018).

Recommendation systems have become fascinating in different practical applications and played a vital role in both academy and industry

* Correspondence to: Dr. Binh Thanh Nguyen, Department of Computer Science, University of Science, Vietnam National University in Ho Chi Minh City, 227, Nguyen Van Cu street, District 5, 700000, Ho Chi Minh City, Viet Nam
E-mail addresses: dachuu25@gmail.com (D.H. Nguyen), huynhthanh1234vn@gmail.com (S.T. Huynh), dinhvietcuong1996@gmail.com (C.V. Dinh), htphong7@gmail.com (P.T. Huynh), ngtbinh@hcmus.edu.vn (B.T. Nguyen).

[1] https://github.com/BinhMisfit/PSRMTE
[2] https://publication-recommender.ieee.org/home.
[3] https://journalsuggester.springer.com/.
[4] https://journalfinder.elsevier.com/.
[5] https://www.keaml.cn:8081/.
[6] http://www.keaml.cn/prs/.

https://doi.org/10.1016/j.eswa.2022.117096
Received 6 June 2021; Received in revised form 15 December 2021; Accepted 28 March 2022
Available online 20 April 2022

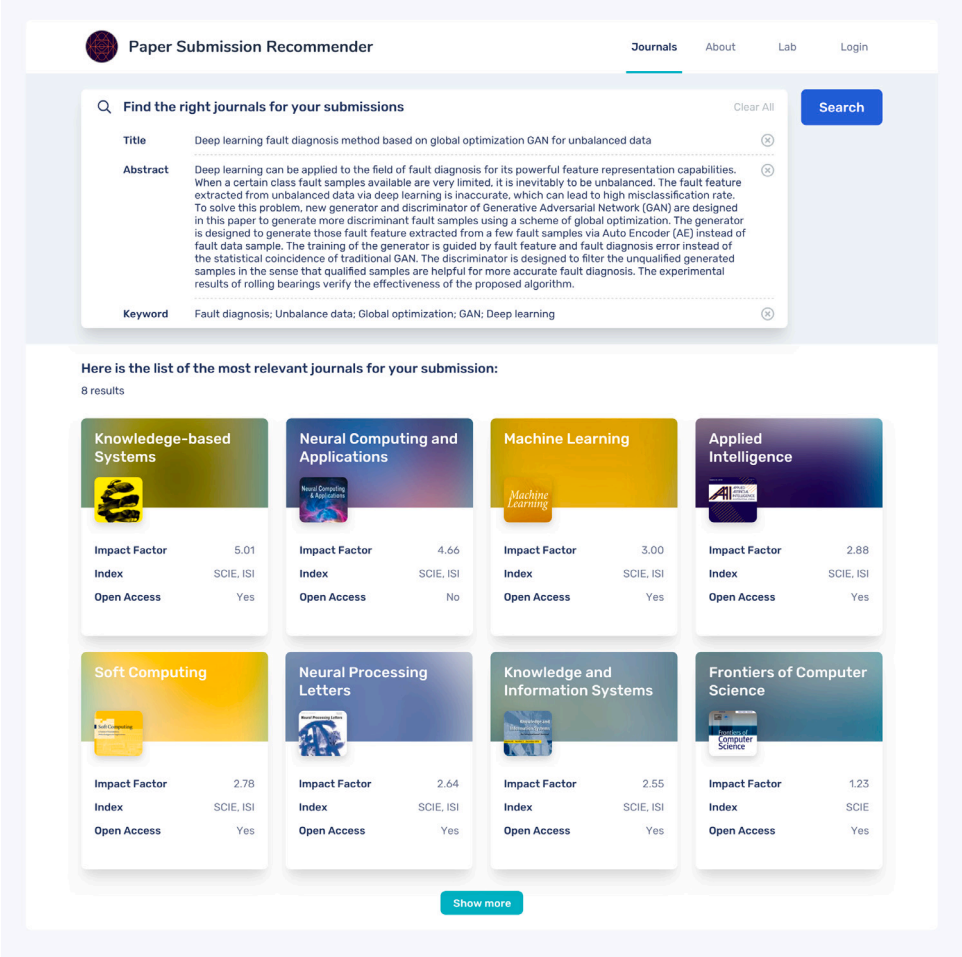

Fig. 1. A paper submission recommendation application for researchers (Cuong et al., 2020).

during the last two decades. It also plays an indispensable part in many business models that can help companies boost their various products and optimize all possible costs as efficiently as possible (Jannach & Jugovac, 2019). Many mega-corporations worldwide (Google, Facebook, Amazon, eBay, and Netflix) have been using mixed recommendation engines as a mainstream industry to maximize their ROI based on tons of information collected from users' preferences, activities, interests, social networks, and hobbies. It is crucial to building intelligent assistance systems to help researchers optimize their works and accelerate research processes. Those are recommending collaborations (Chaiwanarom & Lursinsap, 2015; Liu et al., 2018), papers (Bai et al., 2019), citations (Bhagavatula et al., 2018; Jeong et al., 2020), or publication venues (Feng et al., 2019; Medvet et al., 2014; Pradhan & Pal, 2019; Safa et al., 2018; Wang et al., 2018).

The paper submission recommendation (PSR) problem can be formulated as follows: using a given paper submission's existent information, authors want to find the top N relevant journals and select the most suitable one for the final version. As depicted in Fig. 1, in such a recommendation system, a user only needs to put the title, the abstract, and the list of keywords as the input data and get the appropriate list of top journals recommended. Typically, the input data include three attributes: the title, the abstract, and the list of keywords, and both title and abstract have a word limit. For instance, the abstract of each scientific paper usually has a maximum of between 250 and 300 words, while several journals require that the corresponding abstract has at most 150 words. Therefore, investigating an efficient recommendation engine for the PSR problem is valuable. It can help researchers quickly determine the most relevant journals for submission, especially those studying new research domains, working on interdisciplinary projects, and lacking experience.

A basic idea of constructing a PSR system is leveraging the semantic information from the title, the abstract, and the list of keywords in each paper to extract beneficial features and collect a good enough training dataset to develop a capable recommendation engine. Typically, the training dataset can contain multiple scientific journals, and in each journal, there may be many papers published. For a given submission, the PSR system's primary goal is to calculate the matching score between this submission and each journal in the dataset, based on all papers published in that journal, and return the list of the top suitable ones based on the sorted scores.

Most recently published results use the content-based approach to extract useful features (including TF–IDF, chi-square statistics, and the one-hot encoding technique (Son et al., 2020; Wang et al., 2018). Moreover, those papers trained a recommendation engine to efficiently estimate the matching scores for various journals and return the most proper ones using classical machine learning methods (logistic regression and multi-layer perceptrons). However, few works are related to the problem, and the problem still has challenges for improvement in terms of accuracy and efficiency in various research domains. Especially in the experiment of Wang et al. (2018), the corresponding dataset is not large enough (only having 14,012 papers) for training more complex algorithms while avoiding the overfitting issues. In reality, datasets for the paper recommendation system are usually large and have different kinds of scientific papers (including proceedings or scientific journals).

Up to now, bidirectional transformer encoders have proven their efficiency for improving the performance of multiple state-of-the-art

algorithms in different NLP problems (Vaswani et al., 2017). For this reason, in our study, we want to employ recent bidirectional transformer encoders to investigate if this approach can help to enhance the state-of-the-art related techniques. With this motivation, we have designed the PSR system using a mixture of various transformer encoders and compared the proposed method with other previous techniques. According to our knowledge, this is the first time bidirectional transformer encodes are applied to the paper submission recommendation problem.

To address the challenges mentioned earlier, in this work, we aim to investigate the PRS problem by using advanced deep learning techniques, especially for two popular research domains: computer science and applied mathematics. We consider two datasets for the paper submission recommendation problem, including one dataset collected by Wang and co-workers (Wang et al., 2018) that contains multiple scientific papers in 28 journals and 38 conferences in computer science. We collect another dataset with 223,782 articles among 178 Springer journals in applied mathematics up to early 2020. With given paper submission as the input data, we propose necessary data processing step, and then utilize different bidirectional transformer encoders, including BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), and ELECTRA (Clark et al., 2020), and Mixture of Transformer Encoders (MTE) technique (Jacobs et al., 1991) for constructing an efficient recommendation algorithm for the PRS problem. It is worth noting that those embedding techniques hold state-of-the-art performances on many popular datasets in natural language processing (NLP) like IMDb, Yelp, EBM-NLP, and SciCite. Combining those methods with MTE architecture can create a novel algorithm that can outperform state-of-the-art algorithms for the paper submission recommendation problem with a significant margin, as depicted in our experiments later. Extensive experiments on these two datasets demonstrate the superior performance of our proposed techniques over other previous works.

Finally, the contributions of our work can be summarized as follows:

(a) In the paper submission recommendation problem, we are the first to use bidirectional transformer encoders for constructing an appropriate recommendation engine. The experimental results show that applying these transformer encoders can substantially enhance the performance of the problem.
(b) We incorporate an effective framework that can combine different bidirectional transformer encoders by the Mixture of Transformer Encoders framework into the paper submission recommendation problem. The new algorithm can surpass other methods with a significant margin in both datasets.
(c) We provide all datasets collected for the paper submission recommendation problem, and all codes are available for further interests.[7]

The paper can be organized as follows. Section 2 presents the associated related work of the main problem in two directions: transformer encoders and the recent techniques. We provide the preliminaries for the main problem in Section 3 and show our primary methodology in Section 4. Next, all experimental results and datasets are depicted in Section 5, along with further discussions. Finally, the paper ends with our conclusion and future works.

2. Related work

2.1. Transformer encoders

Before the initial results of Transformers (Vaswani et al., 2017) were proposed in 2019, most natural language processing tasks (such as, e.g., translation and text summarization) had used Recurrent Neural Networks (RNNs) architecture for achieving the most suitable performance (Jordan, 1986; Rumelhart et al., 1986). However, the weakness of this approach is that it is difficult to capture the long-term between words in the sentence, which is called the long-range dependency problem of RNN, and the training speed is usually slow due to sequential input processing. Although Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) is claimed to be able to capture long-range dependency far superior to RNN, it still has another problem. Due to the structure, each unit in the architecture can affect every other due to out-of-memory. Additionally, the LSTM architecture is designed to be processed sequentially that may take a longer processing time.

Transformers were born to solve these challenges. First, a transformer is a non-sequential technique that allows us to process a sentence as a whole rather than word by word; hence, training time and inference time are way faster than LSTM. Secondly, instead of "past information retained through past hidden states" like the Markov property of RNN based (like LSTM or GRU), transformers used self-attention as a different "unit" to compute similarity scores between words in a sentence. And finally, the transformer provides positional embedding using weights to encode information related to the relative position of that word in the string. It is worth noting that the transformer is a deep learning approach that can inherit from the mechanism of attention (Vaswani et al., 2017). Therefore, this approach can weigh the information based on its context and efficiently make the model work. Consequently, it has been mainly used in a massive number of problems in natural language processing. Also, the variations of the transformer architecture (for instance, BERT Devlin et al., 2019 or XLnet Yang et al., 2019) have become the new state-of-the-art techniques. They can process large data while still saving time compared to older architectures.

Many achievements of transformers in a broad range of NLP tasks like text classification, machine translation, and question answering. Among these models, there are multiple popular and significant works like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) and GPT (Generative Pre-trained Transformer) version 1–3 (Brown et al., 2020; Radford & Narasimhan, 2018; Radford et al., 2019). Even that, there is an increasing number of practical applications by transformers in computer vision, including object detection (Carion et al., 2020; Zhu et al., 2020), segmentation (Ye et al., 2019), and super-resolution (Yang et al., 2020) due to their advantages.

2.2. Recent methods

One of the first datasets related to the paper submission recommendation problem was contributed by Klamma et al. (2009). Remarkably, this dataset comprises the datasets from DBLP[8] and Eventseer.[9] Also, Klamma and his colleagues proposed a simple approach by using critical information about participated individuals in similar conferences to build a collaborative filtering model for recommending related scientific events.

Medvet et al. (2014) investigated different approaches for recommending relevant publication venues by using abstract, title, and keywords as the input data and considering three methods: Cavnar Trenkle, two-step Latent Dirichlet Allocation, and the combination between Latent Dirichlet Allocation and K-mean clustering. The paper compared the performance among these techniques on one dataset collecting 58 000 papers in computer science from Microsoft Academic Search[10] that belong to 300 conferences. Luong and colleagues (Luong et al., 2012) studied the scientific venue recommendation problem using co-author networks and social network analysis to extract relevant

[7] https://github.com/BinhMisfit/PSRMTE.

[8] https://dblp.org/.
[9] Eventseer.net.
[10] https://academic.microsoft.com/.

features. The authors compared the proposed methods on one dataset comprising the papers from 16 ACM conferences with 960 documents.

Using the relations between two different authors in academic social graphs, Beierle et al. (2016) proposed six distinct ways for extracting valuable features of the academic conference recommendation problem and using K-nearest neighbor (KNN) (Fix & Hodges, 1989) algorithms for building the most suitable model. Peiris and Weerasinghe (2015) utilized the citation graph extracted from the publication history and citation networks to derive valuable features for the publication venue recommendation problem and propose four different methods (Citation Count, PageRank, Timed PageRank, and Adjusted PageRank). However, similar to Klamma et al. (2009), the paper still used two data sources crawled from DBLP and EventSeer for measuring the performance of proposed techniques compared to the previous methods.

Wang and colleagues (Wang et al., 2018) propose useful features by using the Chi-square and the term frequency-inverse document frequency (TF–IDF) and then choosing the linear logistic regression for building a recommendation model. As a result, they can obtain an accuracy of 61.37% for recommending the most suitable conferences or journals in computer science with a given manuscript. Later, Son and co-workers (Son et al., 2020) surpass the performance of Wang et al. by applying the multiple-layer perceptrons and present additional features extracted from the title, the abstract, and the list of keywords of a given paper, with the highest accuracy (top 3) about 89.07%. Feng et al. (2019) investigated the problem in one large-size dataset with 880,165 papers from 1130 various PubMed Central (PMC) journals. They utilized pre-trained word2vec embeddings and convolutional neural networks to derive essential features from each paper's abstract. Subsequently, they trained a recommendation engine by a fully connected softmax model and integrated it into a publication recommendation system, namely Pubmender.

Pradhan and Pal (2019) introduced a novel scholarly venue recommendation system, namely the DISCOVER system, by combining social network analysis and contextual similarity. They proposed the following features for the recommendation problem: centrality measure calculation, topic modeling based on contextual similarity, citation and co-citation analysis, and key-route identification based primary path analysis of a bibliographic citation network. Their proposed method could outperform other recommendation techniques in the Microsoft Academic Graph (MAG) dataset in various metrics: precision@k, stability, nDCG@k, average venue quality, diversity, accuracy, MRR, and F-scores. They later extended their approach (Pradhan & Pal, 2020) by combining a rank-based fusion of paper-paper peer network and venue-venue peer network models for the final recommendation algorithms. The experiments show the technique could promising results compared to other state-of-the-art methods on the DBLP dataset.

Pradhan and colleagues (Pradhan et al., 2020) continued investigating a venue recommendation system by employing a bi-directional LSTM (Bi-LSTM) and a Hierarchical Attention Network (HAN) to construct the corresponding recommendation model with the following information from the submission: the abstract, the title, the list of keywords, the field of study, and the historical records of the authors. The experimental results on the DBLP-Citation-Network V11 dataset showed that the proposed technique could outperform various previous approaches in F1-score, accuracy, NDCG, MRR, average venue quality, and stability. One can find other related works at (Alshareef et al., 2019; Kim, 2020; Nguyen et al., 2021).

2.3. Relevant scientific paper recommendation

Another approach is related to the paper submission recommendation system, which focuses on recommending the top relevant scientific papers close to a given article. For example, Sakib et al. (2021) and Lu et al. (2021) presented potential methods for recommending related scientific publications from a scientific research paper by using a content-based recommendation (CBR) approach with the input of features like title, abstract, and keywords and collaborative filtering recommendation (CRF) using citation papers as the input data. However, due to the disadvantages of both the CBR and CFR models, they combined these two models. They conducted the corresponding experiments on one dataset, including a list of 50 researchers' publications in the fields consisting of software engineering, programming languages, security, operating systems, networks, information retrieval, graphics, and user interface design.

Yibo Lu and colleagues also used a Multi-layer perceptron (MLP) to enhance recommendation prediction. Adjacent with the CBR and CRF, Li et al. (2021) proposed other paper recommendation methods, called network-based methods. With network-based methods, they said it could full fill the lack of "diversify and novelty" of CBR and "data sparsity and cold start problem" of CRF by building or generating a relation network, such as citation or co-author network. However, these network-based methods request the meta-path (users' historical references) to construct a homogeneous or heterogeneous network, and this type of information is hard to find or get.

Maha Amami and his colleagues (Amami et al., 2016) suggested another method for the researchers to find publications relevant to their research interests. They compared their proposed techniques with the previous ones on the dataset of ArnetMiner.[11] First, Maha Amami utilized an LDA approach to extract the distribution of topics in the researchers' corpus ($vector_1$) and apply a language model for recommendation corpus ($vector_2$). Next, the authors calculated the similarity of $vector_1$ and $vector_2$ by exploiting the symmetrized Kullback Leibler divergence. Also, Ali et al. (2020) gave some enhancement on this problem by using extra features such as papers' citations proximity, authors' information, papers' topical relevance, venues' information, and researchers' preference dynamics. They conducted their experiments on two datasets including DBLP and AAN[11]. For solving the challenge of how to recommend related papers in the same field from a scientific study, Liu and colleagues (Liu et al., 2020) proposed a new keywords-driven and popularity aware approach based on an undirected paper citation graph, named $PR_{keyword+pop}$. Other relevant works in this topic can be found more details at (Cai et al., 2018; Habib & Afzal, 2019; Hong et al., 2013; Zhao et al., 2015).

3. Preliminaries

Before delving into our method and experimental details, we briefly introduce recent transformer techniques such as BERT, XLNET, and ELECTRA. In natural language processing, big models take advantage of available text data to learn the deep contextual meaning of language by optimizing some unsupervised tasks.

3.1. BERT

A typical way of learning contextual embedding distribution is by language modeling, predicting the next word of a given sequence of symbols. This can be done by estimating conditional probabilities over the sequence $P(s_n|s_1, \ldots, s_{n-1})$. This approach only looks at one side (either left or right) of the word and consequently reduces the capability of understanding context from both sides. The BERT creators (Devlin et al., 2019) introduced the Masked Language Model (Masked LM) by replacing a random percentage of input tokens with a special token and then predicting those masked tokens. It can allow the model to learn a bidirectionally contextual representation of input tokens. Still, it can create a mismatch during fine-tuning on downstream tasks where the special mask token does not appear. To tackle this challenge, the authors replaced the masked token with the special token for only 80% of the time and kept the token unchanged for 10% of the time; otherwise, they could replace it with a random token in the vocabulary.

[11] https://aminer.org/citation.

In addition to Masked LM, one can train models for a binary next sentence prediction (NSP) to let models understand the relationship between sentences. Specifically, two sentences are inputted into the model to determine if the two sentences are actual adjacent sentences of the same article or two separate items. Latter research (Lan et al., 2020) argues that NSP is pretty helpful in topic detection, which is close to our current task of predicting the topic of a paper to recommend publication venues. Under the work of Turc et al. (2019), it has been shown that BERT architecture and objectives are effective on a wide range of model sizes, which enables systems with limited computational resources to run in higher performance. Smaller BERT models are most effective under a knowledge distillation setting where one can first train models with more accurate teachers' soft labels before actual fine-tuning.

Moreover, it is undeniable that corpus plays an essential role in pre-trained models' effectiveness on specific downstream tasks. For example, Beltagy et al. (2019) released a version of BERT called SciBERT for the scientific community. The model was trained on a random sample of 1.14 M papers from Semantic Scholar (Cohan et al., 2019). This corpus consists of 18% scientific papers and articles from the computer science domain and 82% from the broad biomedical field. It is true that, to some extent, SciBERT overfits our dataset as some of them are likely to appear in the corpus. However, the SciBERT was trained on different objectives and proved to be good at understanding scientific texts through many benchmarks.

3.2. XLNet

The XLNet authors (Yang et al., 2019) pointed out two weaknesses of BERT:

1. Independence Assumption, i.e., BERT assumes the independence of conditional probabilities among masked tokens, or in other words, all masked tokens are constructed separately.
2. Input Noise: To mask the predicting tokens, BERT uses a special symbol that may never occur in downstream tasks, resulting in a discrepancy between pre-training and fine-tuning.

Consequently, XLNet proposes a permutation language modeling objective based on auto-regressive (AR) models. Let x be a sequence input of length T and Z_T be the set of all possible permutations of $(1, 2, \dots, T)$. Denote z_t as the tth element of a permutation z in Z_T. The training parameters θ of the model are learned to maximize

$$\mathbf{E}_{z \in Z_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | x_{z_1}, x_{z_2}, \dots, x_{z_{t-1}}) \right].$$

The probability $p_\theta(x)$ is decomposed according to the factorization order z. But as a result of optimizing θ over all possible permutation z, in expectation, each token can see every possible other tokens in the sequence and thus capture bidirectional context.

Furthermore, XLNet also uses two-stream self-attention to raise the model's awareness of the target position and Transformer-XL to foster long-range dependencies.

3.3. Electra

Electra (Clark et al., 2020) (which stands for Efficiently Learning an Encoder that Classifies Token Replacements Accurately) proposes a more sample-efficient pre-training task called replaced token detection. Instead of reconstructing the original tokens substituted with special symbols, the model tries to distinguish tokens in the input from tokens replaced by another model. More specifically, at first, a small generator is trained to perform Masked LM in a maximum likelihood rather than an adversarial setting due to the difficulty of applying GANs to text. One can then do the training process for a discriminative model to predict which tokens are original and which ones were corrupted and replaced by the generator. Finally, the discriminator is used as a pre-trained model for fine-tuning on downstream tasks. Electra learning is substantially effective in model sizes, training corpus and outperforms XLNet and BERT at scale on the GLUE natural language understanding benchmark.

4. Methodology

To construct an appropriate recommendation model, we formulate our problem as a classification problem. An expected model should estimate the conditional probability of a given paper submission belonging to a journal with necessary information (title, abstract, and the list of keywords). It can then calculate the matching score between the paper submitted and each journal and then use them to return the top N relevant journals. Consequently, one of our work's main goals is to extract useful features from three attributes of the input data attributes and learn a suitable machine learning model. For better comparison with previous works, including Wang et al. (2018) and Son et al. (2020), we also consider seven different combinations among the title, the abstract, and the list of keywords as the input data of the recommendation problem. Those are abstract (A), title (T), keywords (K), abstract+title (TA), abstract+keywords (AK), title+keywords (TK), and abstract + title + keywords (TAK).

This paper aims to investigate deep learning techniques to extract more meaningful features to improve state-of-the-art algorithms' latest performance. To the best of my knowledge, few studies are using deep-learning features for the PSE problem. In what follows, we present all the main procedures in our approach, including data preprocessing, tokenization, and model architectures used. Notably, we describe how we utilize different bidirectional transformer encoders (e.g., various BERT versions, XLNet, and Electra) and combine those methods using the Mixtures of Transformer Encoders technique to create an efficient recommendation model. Fig. 2 depicts the workflow of using each bidirectional transformer encoder for building the recommendation algorithm, and Fig. 3 presents the MTE framework in our work.

4.1. Data preprocessing

Like other NLP tasks, data preprocessing is essential for the paper submission recommendation problem, especially for noisy and unstandardized data. For having a consistent input format, we always use the lowercase form of input data. It is important to remark that the abstract of a manuscript often contains Latex scripts or mathematical symbols and equations, which cause many difficulties in feature selection. Thus, we remove all phrases likely to be Latex codes or mathematical equations from a given abstract.

4.2. Tokenization

After the data preprocessing step, we break down words in the input text into sub-words (or tokens) to ease the burden of extensive English vocabulary. There are several ways to do tokenization, and it is common to use statistical methods such as Byte Code Encoding (Kudo & Richardson, 2018; Sennrich et al., 2016; Wu et al., 2016). Those methods produce a fixed-size vocabulary and allow the encoding of out-of-vocabulary words as a sequence of tokens. Typically, one can generate a token dictionary by minimizing the number of tokens that a given corpus has after segmentation. Therefore, for each transformer encoder (Bert, SciBert, XLNet, and Electra), we utilize its word tokenizers and vocabulary, previously trained on an open-domain corpus. It is essential to emphasize that contemporary pre-trained NLP models often follow the same input convention as BERT. First, a special token [CLS] is placed at the beginning of the input to indicate that the corresponding output embedding is utilized for classification tasks. Second, special [SEP] tokens allow the model to identify boundaries between different parts of the input data.

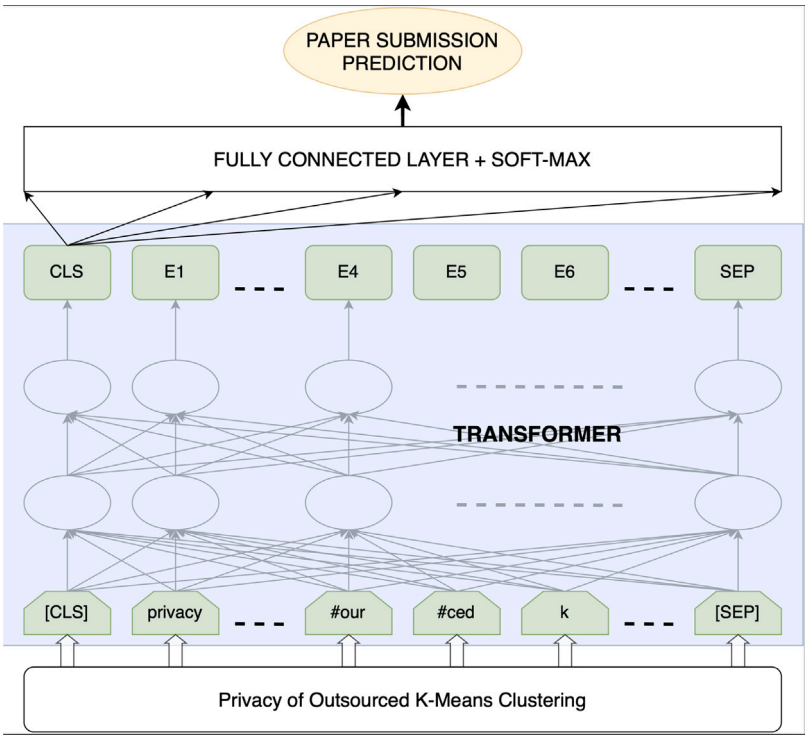

Fig. 2. The proposed workflow of using bidirectional transformer encoders in the PSR problem.

Our experiments concatenate inputs together as a single text and tokenize this text into a sequence of tokens with one [CLS] and one [SEP] at two ends. Moreover, we limit the sequence length for each sample. Any samples containing more tokens than the length limit are trimmed down. We restrict the title or the list of keywords within 128 tokens and the abstract within 512 tokens. The concatenation of both the title and the list of keywords in each paper submitted must have at most 256 tokens. And other combinations that include abstracts must have no more than 512 ones.

4.3. Architectures

Apart from the same standard input, deep learning models in natural language processing often share the same neural network architectures. For example, BERT, XLNet, and Electra are all multi-layer bidirectional transformer encoders sharing the same structure based on the original Transformer implementation in Vaswani et al. (2017). However, XLNet uses TransformerXL (Dai et al., 2019) that can give the capability to learn better with more prolonged dependency. If we denote the number of hidden layers as L and the hidden embedding size as H, the number of self-attention heads and the feed-forward/filter size are always set to $H/64$ and $4H$ respectively. Table 1 shows the architecture of different models with various sizes in our experiments.

To build a proper recommendation algorithm in this work, we consider two scenarios. First, we extract features from the input data using different versions of BERT and both XLNet and Electra. Then we pass them through the last fully connected layer with the Softmax activation function to obtain the recommended output. This architecture can be depicted in Fig. 2. In this figure, the input data include the title of one paper called "Privacy of Outsourced K-Means Clustering". It is tokenized into a sequence of tokens ("CLS", "privacy", "of", "out", "#our", "#ced", "k", "-", "means", "cluster", "#ing", "SEP") before being fed to Transformers. The embedding vector corresponding to the CLS token is later used to predict top relevant journals through a Softmax layer.

Second, as each of these methods has its strength and weaknesses, it is reasonable to take advantage of them. Therefore, we combine three different models, SciBert, XLNet, and Electra, into a unique architecture using the following Mixtures of Transformer Encoders. It is important to emphasize that this is the first time multi-layer bidirectional transformer encoders used for the PSR problem.

4.3.1. Mixtures of transformer encoders

There exist many pre-trained language models, each of which may capture different characteristics of the trained data. Therefore, a practitioner may face the challenge of choosing a suitable pre-trained model for a downstream problem at hand. This problem motivated us to develop a framework to take advantage of many pre-trained language models and unify them into a single learning algorithm. Adapted from the Mixtures of Experts framework (Jacobs et al., 1991), we present the Mixtures of Transformer Encoders (MTE) architecture to create a new recommendation algorithm that outperforms other learning methods.

As visualized in Figs. 3, one can see the corresponding structures of MTE. We use one gating networking to learn suitable weights for selected experts and regard each model mentioned above (SciBERT, XLNet, or Electra) as an expert to make the final recommendation based on the weighted linear combination of the experts.

Let n be the total number of journals and $\overrightarrow{O_1}$, $\overrightarrow{O_2}$, and $\overrightarrow{O_3}$ be the output vectors of the length n from the three trained models, respectively. A gating network G tries to calculate the corresponding weight scalars, G_1, G_2, and G_3, from the input data, x and estimate the predicted distribution overall scientific journals selected as follows:

$$\hat{y} = \text{softmax}\left(G_1 * O_1 + G_2 * O_1 + G_3 * O_1\right),$$

It is important to remark that from the input data x, by doing data preprocessing, one can obtain the embedded vector z by using TF–IDF features (similar to Son et al. (2020)). Using this vector, we compute the weight scalars G_1, G_2, and G_3 by the following formula:

$$z_0 = \text{ReLU}(z \cdot W_0 + b_0)$$
$$G_1 = z_0 \cdot W_1 + b_1$$
$$G_2 = z_0 \cdot W_2 + b_2$$
$$G_3 = z_0 \cdot W_3 + b_3$$

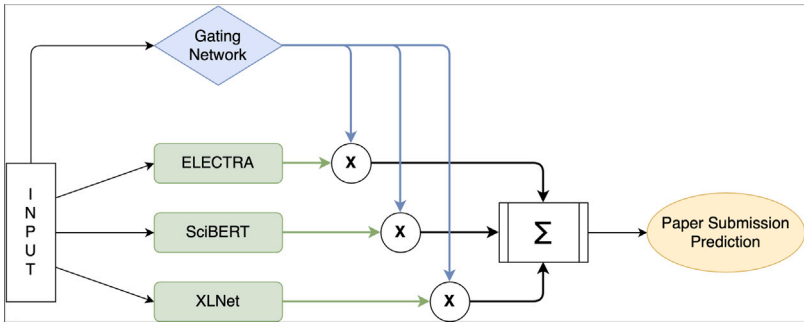

Fig. 3. The Mixtures of Transformer Encoders (MTE) architecture for the PSR problem.

In experiments, we choose W_0, W_1, W_2, and W_3 as vectors of the length 300 and b_1, b_2, and b_3 are scalars that turn G_1, G_2, and G_3 as weight scalars.

In summary, we can present the pipeline of our recommendation system following Algorithm 1.

Algorithm 1 PSRMTE (Paper Submission Recommendation Using Mixtures of Transformer Encoders)

Input: Title (T), Abstract (A), and Keywords (K)
Output: Top K ($K = 1, 3, 5, 10$) recommended venues and journals

A. Feature engineering:
    (i) Pre-processing
        - Lowercase all the input at once .
        - Remove all unnecessary characters or phrases including punctuation, number, symbol or stopwords.
    (ii) Tokenization
        - Concatenate all inputs together to make them as a single combination.[12]
        - Add [CLS] add the beginning and [SEP] at the end of each input.
B. Inference:
    (i) Pass the processed data to the trained classifier for the prediction.
    (ii) Output the top K recommended items.

5. Experiments

We conduct multiple experiments to evaluate all proposed methods for measuring each recommendation model's performance in two different datasets. Then, we report the performance in each scenario by using the top K accuracy ($K = 1, 3, 5, 10$). Here, the top K accuracy represents the percentage of papers where the corresponding journals correctly appear in the list of top K recommended journals generated by a recommendation model.

One can train the proposed models to maximize the likelihood of fitting the training data by minimizing the cross-entropy loss function. We use Adam optimizer and train all models in 10 epochs with a batch size of 128, the initial learning rate of $5 \cdot 10^{-5}$. In our MTE setting, we train each expert and the gating network separately. The gating network aims to determine which experts give a better contribution to the problem by minimizing the cross-entropy loss between $\hat{y}$ and the one-hot target distribution y as well as using the learning rate of 10^{-4}. We conduct all training processes on TPUs provided by Google Colaboratory.[13] In what follows, we will describe more details in two chosen datasets and the experimental results.

[12] This step depends on their order so we have 7 combinations such as T, A, K, T+A, T+K, A+K, and T+A+K.
[13] https://colab.research.google.com/.

5.1. Datasets

There are two datasets for our experiments. The first dataset was initially created by Wang et al. (2018) and then added missing keywords by Son and co-workers (Son et al., 2020). There are 14 012 papers in this dataset, where the training dataset has 9347 samples, and the testing dataset has 4665 records. Each article contains the abstract, the title, the list of existent keywords, and the corresponding journal. Remarkably, the first dataset only has the papers published in top conferences or journals in the computer science domain. Consequently, we name this dataset "CS". We use this dataset as a baseline dataset to compare and demonstrate the performance improvement of our proposed methods.

As Wang's dataset is relatively small, it makes more sense to use another larger dataset (maybe 10–20 times bigger) to construct the most suitable deep learning models for the main problem and then compare the proposed technique's actual performance to others. As a result, we manually collected the second dataset, "Springer", by crawling as many scientific articles in applied mathematics as possible from Springer belonging to 178 journals. This new dataset has 223,782 papers includes:

- id: the name identifies a paper.
- title: the title of a paper.
- authors: the authors directly write papers.
- abstract: the abstract of a paper.
- keywords: the list of keywords of a paper.
- dates: the date of paper that includes Issue date, Publication dates, Received date, Revised date.
- DOI: the Digital object identifier of a paper.
- Journal: the name of the journal to which the paper belongs.
- href: the official website address of a paper.
- year: the year that paper is published.

In this research, we use title, abstract, and keywords as main features to recommend journals relevant to scientific papers and all the remaining features that can be useful for future research. During experiments, we split the dataset into three parts as a .jsonl file: training set, validation set, and test set. We ensure that each journal has at least 100 papers in the training dataset and at least ten articles in the validation and testing sets. Furthermore, to make the performance measurements more realistic, we select all papers published after 2017 into the validation and the testing sets while the earlier ones are put in the training set. Consequently, we have the dataset "Springer" in our research includes the following sets:

- training set: 193 892 papers.
- validation set: 14 945 papers.
- testing set: 14 945 papers.

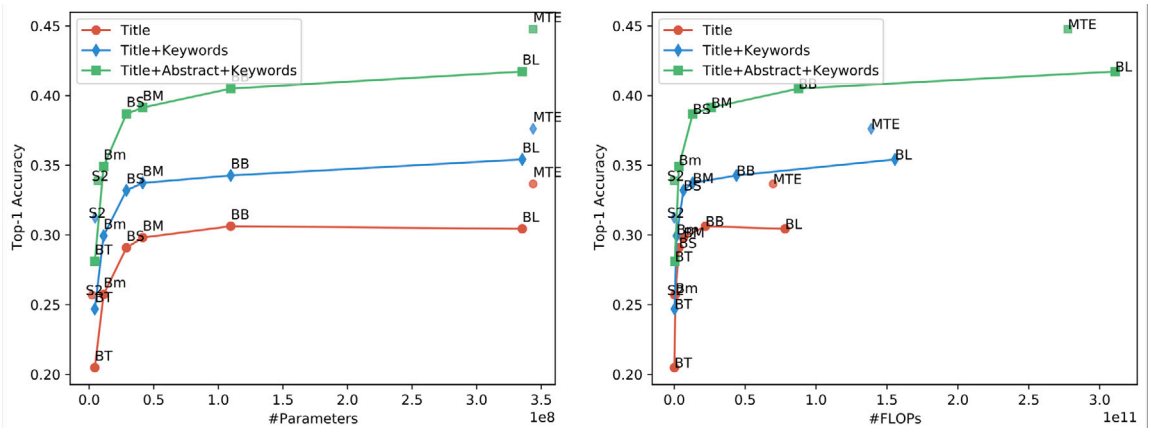

Fig. 4. The performance of different models regarding the number of parameters and FLOPS on the Springer dataset.

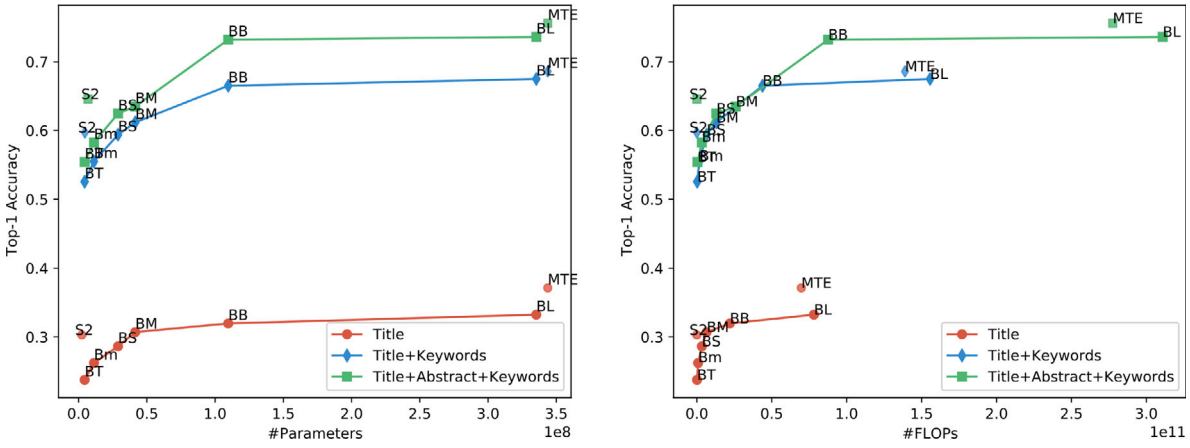

Fig. 5. The performance of different models regarding the number of parameters and FLOPS on the CS dataset.

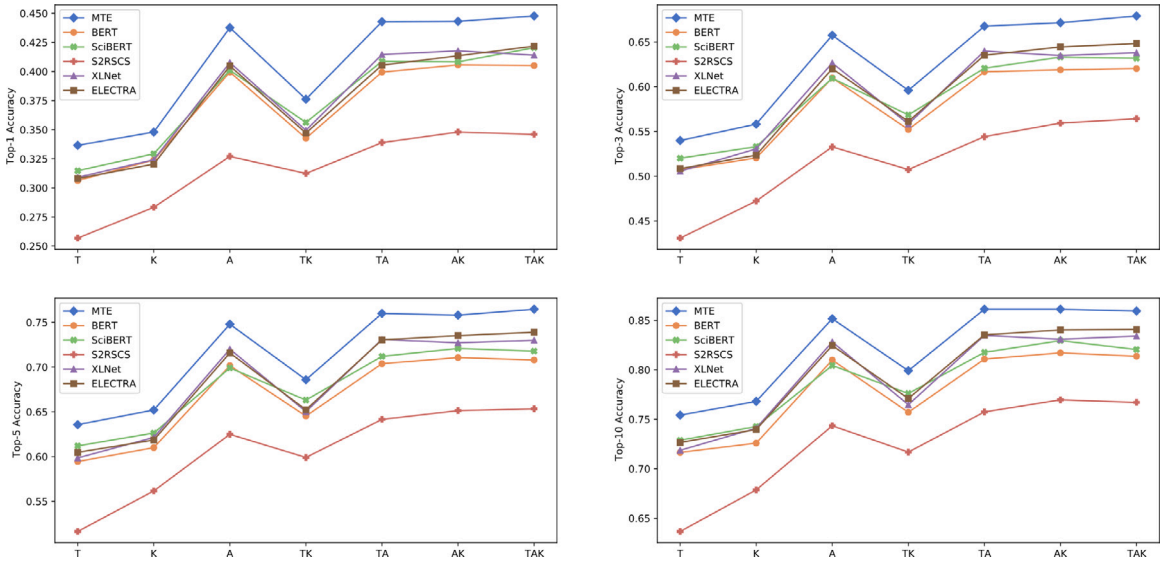

Fig. 6. The performance of different learning models in terms of top K accuracy ($K = 1, 3, 5, 10$) on Springer dataset.

5.2. Experimental results

Using two datasets, as mentioned above, we measure the performance of different bidirectional transformer encoders, S2RSCS, and MTE. Especially, we choose BERT, SciBERT, XLNet, and Electra, each of which consists of 12 layers with the hidden size as 768 in the Transformers architecture. Also, we consider seven types of input data: Title (T), Keywords (K), Abstract (A), Title + Keywords (TK), Title

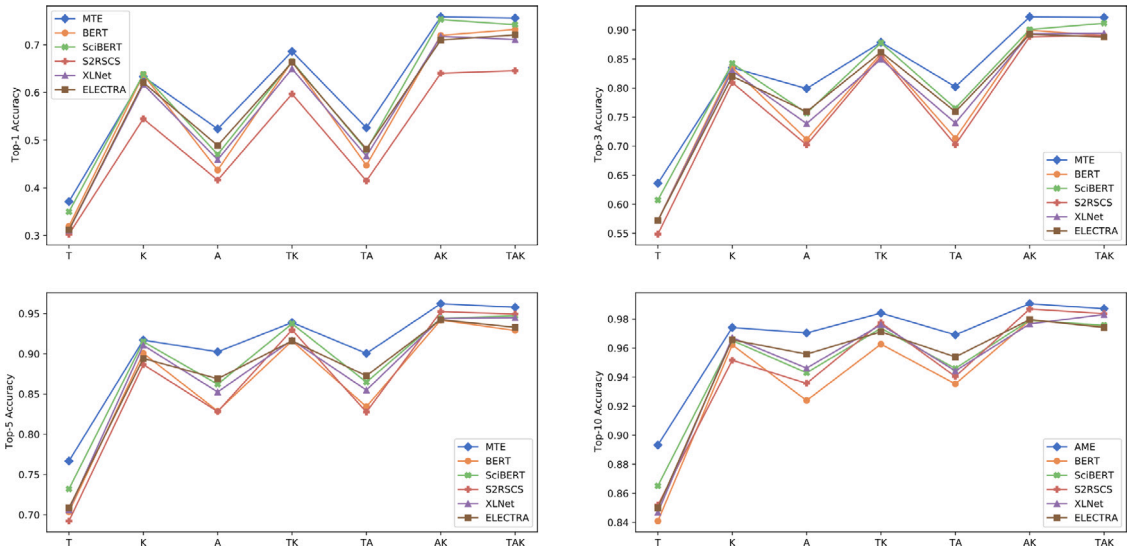

Fig. 7. The performance of different learning models in terms of top K accuracy ($K = 1, 3, 5, 10$) on CS dataset.

Table 1
The configuration of different sizes of learning models used in our experiments.

Model size	L	H
Tiny	2	128
Mini	4	256
Small	4	512
Medium	8	512
Base	12	768
Large	24	1024

+ Abstract (TA), Abstract + Keywords (AK), and Title + Abstract + Keywords (TAK), as well as different sizes of BERT, including $\text{BERT}_{\text{Tiny}}$ (BT), $\text{BERT}_{\text{Mini}}$ (Bm), $\text{BERT}_{\text{Small}}$ (BS), $\text{BERT}_{\text{Medium}}$ (BM), $\text{BERT}_{\text{Base}}$ (BB), and $\text{BERT}_{\text{Large}}$ (BL), to understand the impact of the input size and the model size to the problem. It is worth noting that while all BERT, XLNet, and Electra are trained in an open-domain corpus, SciBERT uses scientific papers in computer science and broad biomedical domain instead.

Figs. 6 and 7 show the performance of different models and input data concerning top K accuracy ($K = 1, 3, 5, 10$). One can see that using T, K, or TK has a smaller performance than using TA, AK, and TAK, respectively, on both two datasets. It turns out that the abstract has a big impact on the paper submission recommendation problem. Using all types of input data (title, abstract, and keywords) can primarily achieve the best performance compared to others in most cases.

On the CS dataset, our proposed method, MTE, substantially outperforms the state-of-the-art algorithm, S2RSCS, in terms of top K accuracy ($K = 1, 3, 5, 10$) for all types of input data. MTE surpasses other techniques presented in this paper except for only using keywords as the input data. In that case, SciBERT is slightly better than MTE for Top 1 accuracy (63.84% vs. 63.34%) and Top 3 accuracy (84.25% vs. 83.51%). Interestingly, the best performance regarding top K accuracy ($K = 1, 3, 5, 10$) occurs when using MTE as a learning model with the input data AK, where we obtain 75.86% in top 1 accuracy, 92.25% in top-3 accuracy, 96.22% in Top 5 accuracy, 99.06% in Top-10 accuracy. Meanwhile, S2RSCS only obtain the best Top 1 accuracy, Top 3 accuracy, Top 5 accuracy, and Top 10 Accuracy are 64.57%, 89.05%, 95.24%, and 98.69%.

Our proposed method, MTE, substantially outperforms all other techniques on the Springer dataset in terms of Top K accuracy ($K = 1, 3, 5, 10$) on both validation and testing datasets. Its best Top 1 accuracy, Top 3 accuracy, Top 5 accuracy, and Top 10 accuracy are 44.77%, 67.90%, 76.46%, and 86.12%, respectively. Remarkably, all bidirectional transformer encoders significantly surpass S2RSCS for all Top K accuracy and input data. For S2RSCS, using Keywords and Abstract in the input data (AK or TAK) can achieve a better performance than other cases, where the best Top 1 accuracy, Top 3 accuracy, Top 5 accuracy, and Top 10 Accuracy are 34.80%, 55.94%, 65.14%, and 76.97%, consecutively. Furthermore, it implies that using new transformer encoders can give an outstanding contribution to the paper submission recommendation.

Moreover, only using Keywords as the input data can have better performance than using Abstract or Title alone on the CS dataset, where the best Top 1 accuracy, Top 3 accuracy, Top 5 accuracy, and Top 10 Accuracy are 63.84%, 84.25%, 91.70%, and 97.42%. However, the phenomenon does not happen on the Springer dataset. One possible reason is that the CS dataset is nearly 20 times smaller than the Springer dataset, so measuring the larger dataset's performance may reflect better than a smaller one.

Among various BERT versions, SciBert performs best compared to other BERTs, including BT, Bm, BS, BM, BB, and BL for both datasets. It may come from the fact that SciBERT is trained in scientific domains while others use open domains. Among SciBERT, XLNet, and Electra, there are no clear trends from the experiment results. SciBERT may marginally surpass others in the CS dataset, but Electra could take the lead in some cases for Top 3, Top 5, and Top 10 accuracy in the Springer dataset.

Finally, we want to assess how each learning model's size can affect the performance of our problem in terms of the number of parameters and FLOPS. Noticeably, FLOPS is the number of floating-point operations (FLOPs) as to how much computational power is needed for a model to predict a single query. We report the comparison by different BERT models with S2RSCS and our best-proposed method, MTE. Figs. 4 and 5 illustrate the results in both two datasets.

In general, S2RSCS is pretty good on a small scale. Despite being small, it can perform well at an accuracy comparable to other deep learning methods of the same resources. TF–IDF vectorization and a single hidden layer neural network in S2RSCS may be as powerful as BERT_{Mini}, which has four hidden layers of 256 nodes in the Transformer. However, it still could not exceed BERT_{Small} or others. On the other hand, the accuracy increases concerning the number of parameters and FLOPS for different BERT versions and MTE. MTE always gets a higher Top 1 Accuracy than BL (nearly $2-3\%$), although its number of parameters is slightly bigger, and its FLOPS is smaller than BL's. Among all BERT models, larger size, better accuracy. One can see more details of our experimental results in Tables 2, 3, 4, 5, 6, 7, and 8.

Table 2
The performance of different learning models in CS dataset with four other types of input data: Title + Abstract, and Title + Keywords. Our proposed method, **MTE**, substantially outperforms all other techniques, including the state-of-the-art algorithm, which is colorized by the red color and proposed by Son et al. (2020), on this dataset in terms of Top N accuracy ($N = 1, 3, 5, 10$).

Input	Model	Top 1	Top 3	Top 5	Top 10
Title Abstract	$BERT_{Tiny}$	0.3659	0.6229	0.7488	0.8727
	$BERT_{Mini}$	0.4081	0.6755	0.7880	0.8952
	$BERT_{Small}$	0.4405	0.7053	0.8047	0.9012
	$BERT_{Medium}$	0.4363	0.7065	0.8246	0.9277
	$BERT_{Base}$	0.4472	0.7136	0.8343	0.9353
	$BERT_{Large}$	0.4786	0.7498	0.8491	0.9349
	SciBERT	0.4789	0.7653	0.8647	0.9460
	ELECTRA	0.4815	0.7593	0.8727	0.9539
	XLNet	0.4667	0.7400	0.8549	0.9443
	MTE	**0.5260**	**0.8021**	**0.9005**	**0.9691**
	S2RSCS	0.4148	0.7027	0.8277	0.9404
Title Keywords	$BERT_{Tiny}$	0.5867	0.7993	0.8743	0.9463
	$BERT_{Mini}$	0.6299	0.8306	0.8937	0.9608
	$BERT_{Small}$	0.6458	0.8366	0.9036	0.9478
	$BERT_{Medium}$	0.6117	0.8374	0.9106	0.9692
	$BERT_{Base}$	0.6652	0.8594	0.9160	0.9627
	$BERT_{Large}$	0.6751	0.8634	0.9300	0.9717
	SciBERT	0.6652	0.8778	0.9374	0.9737
	ELECTRA	0.6632	0.8619	0.9160	0.9712
	$ELECTRA_{Large}$	0.6846	0.8758	0.9265	0.9632
	XLNet	0.6498	0.8495	0.9170	0.9762
	MTE	**0.6860**	**0.8788**	**0.9389**	**0.9841**
	S2RSCS	0.5970	0.8549	0.9297	0.9775

Table 3
The performance of different learning models in CS dataset with three types of input data: Title, Abstract, and Keywords. Our proposed method, MTE, substantially outperforms the state-of-the-art algorithm, which is colorized by the red color and proposed by Son et al. (2020), on this dataset in terms of Top N accuracy ($N = 1, 3, 5, 10$). Also, MTE surpasses other techniques presented in this paper except for the case the input data only contain keywords, where SciBERT has a bit better than MTE for Top 1 accuracy and Top 3 accuracy.

Input	Model	Top 1	Top 3	Top 5	Top 10
Title	$BERT_{Tiny}$	0.2557	0.4819	0.6111	0.7655
	$BERT_{Mini}$	0.2722	0.5025	0.6364	0.7820
	$BERT_{Small}$	0.2984	0.5430	0.6757	0.8107
	$BERT_{Medium}$	0.3070	0.5529	0.6872	0.8278
	$BERT_{Base}$	0.3196	0.5721	0.7035	0.8409
	$BERT_{Large}$	0.3324	0.5899	0.7173	0.8538
	SciBERT	0.3498	0.6075	0.7320	0.8652
	ELECTRA	0.3121	0.5721	0.7087	0.8499
	XLNet	0.3113	0.5721	0.7061	0.8469
	MTE	**0.3713**	**0.6362**	**0.7668**	**0.8932**
	S2RSCS	0.3029	0.5486	0.6922	0.8519
Abstract	$BERT_{Tiny}$	0.3678	0.5728	0.6725	0.7891
	$BERT_{Mini}$	0.4373	0.6765	0.7666	0.8722
	$BERT_{Small}$	0.4418	0.7085	0.8118	0.9046
	$BERT_{Medium}$	0.4238	0.7093	0.8236	0.9245
	$BERT_{Base}$	0.4373	0.7117	0.8289	0.9239
	$BERT_{Large}$	0.4693	0.7400	0.8473	0.9333
	SciBERT	0.4701	0.7569	0.8622	0.9430
	ELECTRA	0.4887	0.7593	0.8690	0.9558
	XLNet	0.4592	0.7389	0.8525	0.9462
	MTE	**0.5237**	**0.7991**	**0.9025**	**0.9704**
	S2RSCS	0.4163	0.7023	0.8283	0.9357
Keywords	$BERT_{Tiny}$	0.5563	0.7878	0.8812	0.9463
	$BERT_{Mini}$	0.5941	0.8122	0.8843	0.9399
	$BERT_{Small}$	0.6279	0.8331	0.9026	0.9593
	$BERT_{Medium}$	0.6294	0.8271	0.9056	0.9603
	$BERT_{Base}$	0.6374	0.8351	0.9006	0.9622
	$BERT_{Large}$	0.6354	0.8366	0.9136	0.9632
	SciBERT	**0.6384**	**0.8425**	0.9151	0.9652
	ELECTRA	0.6210	0.8202	0.8942	0.9657
	XLNet	0.6170	0.8296	0.9106	0.9672
	MTE	0.6334	0.8351	**0.9170**	**0.9742**
	S2RSCS	0.5449	0.8096	0.8866	0.9516

Table 4
The performance of different learning models in CS dataset with four other types of input data: Abstract + Keywords, and Title + Abstract + Keywords. Our proposed method, MTE, substantially outperforms all other techniques, including the state-of-the-art algorithm, which is colorized by the red color and proposed by Son et al. (2020), on this dataset in terms of Top N accuracy ($N = 1, 3, 5, 10$).

Input	Model	Top 1	Top 3	Top 5	Top 10
Abstract Keywords	$BERT_{Tiny}$	0.6185	0.8361	0.8972	0.9583
	$BERT_{Mini}$	0.6294	0.8231	0.8857	0.9488
	$BERT_{Small}$	0.7094	0.8818	0.9220	0.9578
	$BERT_{Medium}$	0.7134	0.8803	0.9200	0.9752
	$BERT_{Base}$	0.7198	0.8997	0.9419	0.9791
	$BERT_{Large}$	0.7303	0.8972	0.9434	0.9791
	SciBERT	0.7531	0.9006	0.9439	0.9791
	ELECTRA	0.7099	0.8927	0.9424	0.9796
	XLNet	0.7178	0.8932	0.9439	0.9767
	MTE	**0.7586**	**0.9225**	**0.9622**	**0.9906**
	S2RSCS	0.6403	0.8883	0.9524	0.9869
Title Abstract Keywords	$BERT_{Tiny}$	0.6344	0.8371	0.9021	0.9523
	$BERT_{Mini}$	0.6518	0.8395	0.9101	0.9588
	$BERT_{Small}$	0.7114	0.8932	0.9309	0.9682
	$BERT_{Medium}$	0.7009	0.8892	0.9349	0.9781
	$BERT_{Base}$	0.7322	0.8902	0.9290	0.9752
	$BERT_{Large}$	0.7362	0.9086	0.9478	0.9871
	SciBERT	0.7422	0.9116	0.9473	0.9757
	ELECTRA	0.7208	0.8877	0.9329	0.9742
	XLNet	0.7109	0.8942	0.9449	0.9831
	MTE	**0.7561**	**0.9215**	**0.9578**	**0.9871**
	S2RSCS	0.6457	0.8905	0.9494	0.9837

5.3. LDA-based approach

This section presents another experiment by comparing our proposed methods with an LDA-based approach.

We preprocess our datasets by converting all words to lowercase for constructing the corresponding LDA-based model. Next, we remove all punctuation marks, emojis, numbers, stopwords, and short words. After that, we convert (or normalize) all the words into their root form using lemmatization. Finally, Bigrams and Trigrams are utilized to diversify the semantics of our data, including three main input attributes (Title, Abstract, and Keywords). We use LDA, where the number of topics is 20 as a feature extractor for each feature combination, including T, A, K, TA, TK, and TAK. After that, we utilize three models respectively, including Logistic Regression (LR) (Cramer, 2002), Random Forest (RF) (Cutler et al., 2011), and Multi-Layer Perceptron (MLP) (Marius et al., 2009) for paper submission recommendation system.

One can see in detail the result of each model in term of Accuracy Top1, Top3, Top5, and Top10 in Tables 9, 10, and 11. The experiments show that all three machine learning models using LDA as feature extraction give a lower performance in terms of accuracy Top1, Top3, Top5, and Top10 compared to both our results and those of Wang et al. (2018). One possible reason to explain this result is that LDA is an unsupervised learning model. Because of that, this technique may take lots of noisy data and cannot take advantage of short meaningful features (the labels or keywords) like TFIDF, or the hidden semantic matrices of deep learning models. Remarkably, those methods used the data's labels to refit the model with our approach.

Interestingly, among all three models, including LR, RF, and MLP, one can gain the best result when using feature Keywords in Top1, Top3, Top5, and Top10. The LDA algorithm may not extract any more information from the list of keywords as keywords carry a dense amount of information about an article. Also, applying LDA to other features such as Title and Abstract can backfire by extracting noisy information. More details can be found in Tables 9, 10, and 11.

5.4. Further discussions

This paper aims to focus on investigating the most suitable algorithm for the paper submission recommendation problem. One of

Table 5
The performance of different learning models in Springer dataset with three types of input data: Title, Abstract, and Keywords. Our proposed method, MTE, substantially outperforms all other techniques in terms of Top N accuracy ($N = 1, 3, 5, 10$) on both validation and testing datasets.

Input	Model	Valid				Test			
		Top 1	Top 3	Top 5	Top 10	Top 1	Top 3	Top 5	Top 10
Title	$BERT_{Tiny}$	0.2047	0.3672	0.4485	0.5617	0.2048	0.3641	0.4466	0.5613
	$BERT_{Mini}$	0.2594	0.4429	0.5311	0.6507	0.2573	0.4393	0.5253	0.6503
	$BERT_{Small}$	0.2953	0.4917	0.5857	0.7054	0.2908	0.4860	0.5819	0.7018
	$BERT_{Medium}$	0.2987	0.4956	0.5902	0.7102	0.2981	0.4990	0.5900	0.7083
	$BERT_{Base}$	0.3069	0.5052	0.6001	0.7175	0.3063	0.5073	0.5945	0.7165
	$BERT_{Large}$	0.3113	0.4835	0.5618	0.6769	0.3044	0.4818	0.5602	0.6707
	SciBERT	0.3182	0.5208	0.6153	0.7378	0.3148	0.5201	0.6120	0.7288
	ELECTRA	0.3101	0.5133	0.6091	0.7308	0.3081	0.5085	0.6048	0.7266
	XLNet	0.3091	0.5091	0.6072	0.7312	0.3091	0.5056	0.5985	0.7186
	MTE	**0.3371**	**0.5460**	**0.6422**	**0.7608**	**0.3366**	**0.5399**	**0.6357**	**0.7542**
	S2RSCS	0.2548	0.4302	0.5153	0.6379	0.2568	0.4308	0.5164	0.6366
Abstract	$BERT_{Tiny}$	0.2643	0.4554	0.5441	0.6513	0.2673	0.4515	0.5434	0.6484
	$BERT_{Mini}$	0.3376	0.5448	0.6336	0.7499	0.3360	0.5409	0.6312	0.7441
	$BERT_{Small}$	0.3750	0.5952	0.6869	0.8004	0.3768	0.5911	0.6829	0.7967
	$BERT_{Medium}$	0.3805	0.5975	0.6917	0.8024	0.3790	0.5969	0.6870	0.7987
	$BERT_{Base}$	0.3985	0.6130	0.7066	0.8160	0.3995	0.6098	0.7016	0.8099
	$BERT_{Large}$	0.4008	0.5894	0.6658	0.7684	0.4002	0.5818	0.6584	0.7619
	SciBERT	0.4097	0.6184	0.7078	0.8163	0.4011	0.6094	0.6991	0.8043
	ELECTRA	0.4074	0.6337	0.7236	0.8293	0.4048	0.6201	0.7158	0.8245
	XLNet	0.4058	0.6335	0.7267	0.8312	0.4077	0.6266	0.7200	0.8282
	MTE	**0.4391**	**0.6662**	**0.7526**	**0.8575**	**0.4377**	**0.6575**	**0.7479**	**0.8516**
	S2RSCS	0.3288	0.5359	0.6316	0.7460	0.3271	0.5328	0.6249	0.7435
Keywords	$BERT_{Tiny}$	0.2335	0.3870	0.4692	0.5782	0.2245	0.3835	0.4601	0.5661
	$BERT_{Mini}$	0.2792	0.4612	0.5457	0.6602	0.2772	0.4519	0.5387	0.6561
	$BERT_{Small}$	0.3081	0.5092	0.5987	0.7154	0.3065	0.5036	0.5957	0.7150
	$BERT_{Medium}$	0.3154	0.5160	0.6072	0.7263	0.3111	0.5122	0.6029	0.7200
	$BERT_{Base}$	0.3289	0.5301	0.6209	0.7427	0.3240	0.5204	0.6100	0.7259
	$BERT_{Large}$	0.3275	0.5227	0.6142	0.7382	0.3211	0.5159	0.6100	0.7274
	SciBERT	0.3385	0.5394	0.6345	0.7530	0.3293	0.5331	0.6264	0.7427
	ELECTRA	0.3218	0.5333	0.6270	0.7465	0.3204	0.5236	0.6185	0.7396
	XLNet	0.3305	0.5348	0.6298	0.7472	0.3242	0.5305	0.6214	0.7410
	MTE	**0.3529**	**0.5644**	**0.6594**	**0.7785**	**0.3480**	**0.5582**	**0.6521**	**0.7681**
	S2RSCS	0.2888	0.4775	0.5738	0.6913	0.2834	0.4723	0.5617	0.6786

Table 6
The performance of different learning models in Springer dataset with three types of input data: Title + Abstract, Title + Keywords, Abstract + Keywords, and Title + Abstract + Keywords. Our proposed method, MTE, substantially outperforms all other techniques in terms of Top N accuracy ($N = 1, 3, 5, 10$) on both validation and testing datasets.

Input	Model	Valid				Test			
		Top 1	Top 3	Top 5	Top 10	Top 1	Top 3	Top 5	Top 10
Title	$BERT_{Tiny}$	0.2643	0.4554	0.5441	0.6513	0.2719	0.4586	0.5508	0.6529
Abstract	$BERT_{Mini}$	0.3421	0.5517	0.6419	0.7531	0.3437	0.5506	0.6424	0.7543
	$BERT_{Small}$	0.3818	0.5953	0.6935	0.8090	0.3889	0.6051	0.6927	0.8017
	$BERT_{Medium}$	0.3866	0.6101	0.7043	0.8099	0.3894	0.6066	0.6958	0.8040
	$BERT_{Base}$	0.4042	0.6252	0.7148	0.8173	0.3995	0.6167	0.7038	0.8108
	$BERT_{Large}$	0.4167	0.5972	0.6778	0.7782	0.4092	0.5926	0.6752	0.7744
	SciBERT	0.4169	0.6319	0.7211	0.8260	0.4088	0.6207	0.7119	0.8176
	ELECTRA	0.4117	0.6432	0.7366	0.8423	0.4055	0.6352	0.7303	0.8353
	XLNet	0.4201	0.6473	0.7370	0.8424	0.4147	0.6399	0.7309	0.8347
	MTE	**0.4476**	**0.6771**	**0.7665**	**0.8674**	**0.4428**	**0.6676**	**0.7599**	**0.8612**
	S2RSCS	0.3374	0.5508	0.6455	0.7627	0.3389	0.5442	0.6416	0.7574
Title	$BERT_{Tiny}$	0.2329	0.3914	0.4732	0.5832	0.2469	0.4126	0.4965	0.6079
Keyword	$BERT_{Mini}$	0.3042	0.4962	0.5813	0.6964	0.2994	0.4915	0.5785	0.6974
	$BERT_{Small}$	0.3388	0.5440	0.6312	0.7465	0.3320	0.5386	0.6289	0.7428
	$BERT_{Medium}$	0.3396	0.5457	0.6385	0.7578	0.3373	0.5386	0.6325	0.7478
	$BERT_{Base}$	0.3542	0.5659	0.6572	0.7690	0.3427	0.5522	0.6453	0.7571
	$BERT_{Large}$	0.3585	0.5411	0.6161	0.7287	0.3542	0.5342	0.6149	0.7182
	SciBERT	0.3609	0.5759	0.6724	0.7849	0.3562	0.5685	0.6632	0.7761
	ELECTRA	0.3453	0.5675	0.6604	0.7784	0.3471	0.5613	0.6524	0.7710
	XLNet	0.3520	0.5584	0.6570	0.7762	0.3495	0.5582	0.6503	0.7647
	MTE	**0.3821**	**0.5972**	**0.6917**	**0.8073**	**0.3762**	**0.5959**	**0.6859**	**0.7991**
	S2RSCS	0.3127	0.5147	0.6107	0.7321	0.3124	0.5075	0.5991	0.7168

(*continued on next page*)

the essential goals is implementing a practical system for the paper recommendation problem by integrating the proposed method.

Up to now, there have been several similar platforms existed in the market, which can be summarized as follows:

- The current paper submission recommendation system by Springer[14] uses both title and abstract as the input, and the

[14] https://journalsuggester.springer.com/.

Table 6 (*continued*).

Input	Model	Valid				Test			
		Top 1	Top 3	Top 5	Top 10	Top 1	Top 3	Top 5	Top 10
Keyword	$BERT_{Tiny}$	0.2833	0.4645	0.5542	0.6604	0.2809	0.4644	0.5528	0.6572
Abstract	$BERT_{Mini}$	0.3512	0.5564	0.6513	0.7605	0.3507	0.5538	0.6401	0.7520
	$BERT_{Small}$	0.3841	0.6042	0.6923	0.8024	0.3853	0.6010	0.6905	0.8023
	$BERT_{Medium}$	0.3902	0.6093	0.6978	0.8051	0.3879	0.6082	0.6958	0.8046
	$BERT_{Base}$	0.4078	0.6225	0.7131	0.8189	0.4057	0.6189	0.7106	0.8171
	$BERT_{Large}$	0.4181	0.6027	0.6787	0.7736	0.4096	0.5964	0.6756	0.7738
	SciBERT	0.4182	0.6401	0.7319	0.8392	0.4083	0.6331	0.7208	0.8295
	ELECTRA	0.4168	0.6449	0.7394	0.8446	0.4135	0.6444	0.7351	0.8403
	XLNet	0.4193	0.6405	0.7303	0.8341	0.4178	0.6348	0.7271	0.8309
	MTE	**0.4481**	**0.6788**	**0.7671**	**0.8680**	**0.4431**	**0.6715**	**0.7580**	**0.8612**
	S2RSCS	0.3528	0.5663	0.6610	0.7788	0.3480	0.5594	0.6514	0.7697
Title	$BERT_{Tiny}$	0.2808	0.4696	0.5573	0.6582	0.2810	0.4726	0.5584	0.6644
Keyword	$BERT_{Mini}$	0.3551	0.5589	0.6473	0.7625	0.3492	0.5530	0.6438	0.7532
Abstract	$BERT_{Small}$	0.3909	0.6092	0.6990	0.8091	0.3869	0.6015	0.6937	0.8035
	$BERT_{Medium}$	0.3965	0.6205	0.7092	0.8163	0.3914	0.6087	0.6985	0.8061
	$BERT_{Base}$	0.4113	0.6240	0.7142	0.8160	0.4051	0.6203	0.7079	0.8137
	$BERT_{Large}$	0.4240	0.6064	0.6828	0.7762	0.4173	0.6025	0.6814	0.7792
	SciBERT	0.4256	0.6379	0.7247	0.8297	0.4205	0.6318	0.7178	0.8205
	ELECTRA	0.4240	0.6539	0.7453	0.8486	0.4218	0.6482	0.7388	0.8407
	XLNet	0.4268	0.6480	0.7392	0.8413	0.4141	0.6379	0.7299	0.8341
	MTE	**0.4579**	**0.6866**	**0.7738**	**0.8700**	**0.4477**	**0.6790**	**0.7646**	**0.8594**
	S2RSCS	0.3527	0.5697	0.6618	0.7777	0.3460	0.5642	0.6534	0.7671

Table 7
The number of parameters and Flops of different learning models and input data in CS dataset.

Input	Model	Parameters ($\times 10^3$)	FLOPs ($\times 10^6$)	Top-1 accuracy
Title	$BERT_{Tiny}$	4411	112	0.2376
	$BERT_{Mini}$	11 221	836	0.2621
	$BERT_{Small}$	28 864	3296	0.2863
	$BERT_{Medium}$	41 474	6557	0.3070
	$BERT_{Base}$	109 633	22 033	0.3196
	$BERT_{Large}$	335 343	78 164	0.3324
	MTE	343 735	69 748	0.3713
	S2RSCS	2354	5	0.3029
Title	$BERT_{Tiny}$	4411	215	0.5255
Keyword	$BERT_{Mini}$	11 221	1653	0.5549
	$BERT_{Small}$	28 864	6539	0.5938
	$BERT_{Medium}$	41 474	13 044	0.6117
	$BERT_{Base}$	109 633	43 874	0.6652
	$BERT_{Large}$	335 343	155 648	0.6751
	MTE	343 735	138 896	0.6860
	S2RSCS	4632	9	0.5970
Title	$BERT_{Tiny}$	4411	426	0.5542
Keyword	$BERT_{Mini}$	11 221	3297	0.5825
Abstract	$BERT_{Small}$	28 864	13 046	0.6245
	$BERT_{Medium}$	41 474	26 056	0.6349
	$BERT_{Base}$	109 633	87 643	0.7322
	$BERT_{Large}$	335 343	310 927	0.7362
	MTE	343 735	277 570	0.7561
	S2RSCS	6892	14	0.6457

output includes relevant journals. In addition, the system can also give users additional information about the journal, such as the minimum impact factor, the minimum acceptance rate, and the maximum time to the first decision.

- Meanwhile, the paper submission recommendation constructed by IEEE[15] only allows users two options: (1) they insert keywords, key phrases, or article titles; (2) otherwise, they can upload their submission files so that the system can extract the corresponding list of keywords or the article titles. However, users could not choose both options to get the recommendation results. However, this system can suggest journals, conferences, or both of them.
- The corresponding paper submission recommendation platform by Elsevier[16] uses both the title, the abstract, and the keywords of each manuscript as the input. However, the output only consists of journals without conferences; similar to Springer, this system also uses publication type, review, publication time, and journal impact.
- Two remaining systems of Keaml[17,18] only allow to insert the abstract of each submission. However, these systems do not filter the recommended results by given research fields.

Remarkably, one can see that these systems have different inputs and contribute to various purposes related to the publishers. However, each platform has certain shortcomings, such as the system of Springer does not use keywords that are critical information for a paper submission. Furthermore, the inputs of IEEE are extremely strict, and the user cannot be flexible in using both title and abstract as the input. Moreover, the platform of Elsevier can have the above features, but it lacks the feature to suggest both conferences and journals for users. Finally, the two systems of Keaml only use abstract, and users cannot choose the field they think suits their paper.

In this work, we focus on investigating the recommendation model for the main problem. By using the title, the abstract, and the list of keywords, the proposed algorithm can suggest the most relevant journals or conferences, or both of them for users. One of our next steps is to integrate the proposed method into a practical system that can support recommended results in different research topics for the research community.

6. Conclusion and future works

We have presented a practical approach for the paper recommendation problem using various bidirectional transformer encoders (different versions of BERT, SciBERT, XLNet, Electra) and the Mixtures of Transformer Encoders framework. We measure other proposed learning models' performance by top K accuracy ($K = 1, 3, 5, 10$) on two datasets: one dataset from Wang et al. including 14 012 papers in computer science, and another dataset collected by us having 223,782 articles in 178 Springer applied mathematics journals. The experimental results show that using transformer encoders can significantly

[15] https://publication-recommender.ieee.org/home.

[16] https://journalfinder.elsevier.com/.

[17] https://www.keaml.cn:8081/.

[18] http://www.keaml.cn/prs/.

Table 8
The number of parameters and Flops of different learning models and input data in Springer dataset.

Input	Model	Parameters ($\times 10^3$)	FLOPs ($\times 10^6$)	Top-1 accuracy (Valid)	Top-1 accuracy (Test)
Title	$BERT_{Tiny}$	4411	112	0.2376	0.2048
	$BERT_{Mini}$	11 221	836	0.2621	0.2573
	$BERT_{Small}$	28 864	3296	0.2863	0.2908
	$BERT_{Medium}$	41 474	6557	0.3070	0.2981
	$BERT_{Base}$	109 633	22 033	0.3196	0.3063
	$BERT_{Large}$	335 343	78 164	0.3324	0.3044
	MTE	343 735	69 748	0.3713	0.3366
	S2RSCS	2354	5	0.3029	0.2568
Title	$BERT_{Tiny}$	4411	215	0.5255	0.2469
Keyword	$BERT_{Mini}$	11 221	1653	0.5549	0.2994
	$BERT_{Small}$	28 864	6539	0.5938	0.3320
	$BERT_{Medium}$	41 474	13 044	0.6117	0.3373
	$BERT_{Base}$	109 633	43 874	0.6652	0.3427
	$BERT_{Large}$	335 343	155 648	0.6751	0.3542
	MTE	343 735	138 896	0.6860	0.3762
	S2RSCS	4632	9	0.5970	0.3124
Title	$BERT_{Tiny}$	4411	426	0.5542	0.2810
Keyword	$BERT_{Mini}$	11 221	3297	0.5825	0.3492
Abstract	$BERT_{Small}$	28 864	13 046	0.6245	0.3869
	$BERT_{Medium}$	41 474	26 056	0.6349	0.3914
	$BERT_{Base}$	109 633	87 643	0.7322	0.4051
	$BERT_{Large}$	335 343	310 927	0.7362	0.4173
	MTE	343 735	277 570	0.7561	0.4477
	S2RSCS	6892	14	0.6457	0.3460

Table 9
The performance of LDA-based techniques using Logistic Regression in CS dataset with seven other types of input data: Title + Abstract, Title + Keywords, Title, Abstract, Keywords, Abstract + Keywords, and Title + Abstract + Keywords in terms of Top N accuracy (N = 1,3,5,10).

	Top1	Top3	Top5	Top10
TA	0.2845	0.5648	0.7260	0.8883
TK	0.3194	0.5805	0.7106	0.8605
T	0.1040	0.2480	0.3681	0.5479
A	0.2776	0.5496	0.7014	0.8682
K	0.3256	0.6026	0.7387	0.8686
AK	0.3293	0.6204	0.7601	0.9130
TAK	0.3106	0.5942	0.7464	0.9025

Table 10
The performance of LDA-based techniques using Random Forest in CS dataset with seven other types of input data: Title + Abstract, Title + Keywords, Title, Abstract, Keywords, Abstract + Keywords, and Title + Abstract + Keywords in terms of Top N accuracy (N = 1,3,5,10).

	Top1	Top3	Top5	Top10
TA	0.2746	0.5496	0.7080	0.8617
TK	0.3258	0.5460	0.6557	0.7961
T	0.0986	0.2384	0.3460	0.5293
A	0.2757	0.5395	0.6787	0.8467
K	0.3741	0.5951	0.6969	0.8221
AK	0.3179	0.5923	0.7368	0.8887
TAK	0.3005	0.5796	0.7303	0.8870

Table 11
The performance of LDA-based techniques using Multi-Layer Perceptron in CS dataset with seven other types of input data: Title + Abstract, Title + Keywords, Title, Abstract, Keywords, Abstract + Keywords, and Title + Abstract + Keywords in terms of Top N accuracy (N = 1,3,5,10).

	Top1	Top3	Top5	Top10
TA	0.3194	0.6116	0.7623	0.9093
TK	0.3393	0.5989	0.7280	0.8707
T	0.1059	0.2497	0.3595	0.5486
A	0.2707	0.5606	0.7117	0.8787
K	0.3558	0.6180	0.7520	0.8817
AK	0.3436	0.6274	0.7728	0.9213
TAK	0.3194	0.6116	0.7623	0.9093

contribute to improving the paper submission recommendation problems. Our proposed method outperforms the state-of-the-art techniques with a significant margin in all top K accuracy.

In future work, we aim to implement a web application for recommending users submit an appropriate journal based on the title, the abstract, and the list of keywords from their submitted manuscript. We are also continuously improving our models by applying other new recommendation algorithms to this problem. Finally, we are going to extend the datasets with the same research topic but across different publishers.

CRediT authorship contribution statement

Dac Huu Nguyen: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Son Thanh Huynh:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Cuong Viet Dinh:** Conceptualization, Methodology, Investigation, Writing – original draft. **Phong Tan Huynh:** Conceptualization, Investigation, Writing – original draft. **Binh Thanh Nguyen:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We want to thank the University of Science, Vietnam National University in Ho Chi Minh City, and AISIA Research Lab in Vietnam for supporting us throughout this paper. This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number C2021-18-03.

References

Ali, Z., Qi, G., Muhammad, K., Ali, B., & Abro, W. A. (2020). Paper recommendation based on heterogeneous network embedding. *Knowledge-Based Systems*, *210*, Article 106438. http://dx.doi.org/10.1016/j.knosys.2020.106438, URL: https://www.sciencedirect.com/science/article/pii/S0950705120305670.

Alshareef, A. M., Alhamid, M. F., & El Saddik, A. (2019). Academic venue recommendations based on similarity learning of an extended nearby citation network. *IEEE Access*, *7*, 38813–38825. http://dx.doi.org/10.1109/ACCESS.2019.2906106.

Amami, M., Pasi, G., Stella, F., & Faiz, R. (2016). An LDA-based approach to scientific paper recommendation. In E. Métais, F. Meziane, M. Saraee, V. Sugumaran, & S. Vadera (Eds.), *Natural language processing and information systems* (pp. 200–210). Cham: Springer International Publishing.

Bai, X., Wang, M., Lee, I., Yang, Z., Kong, X., & Xia, F. (2019). Scientific paper recommendation: A survey. *IEEE Access*, *7*, 9324–9339. http://dx.doi.org/10.1109/ACCESS.2018.2890388.

Beierle, F., Tan, J., & Grunert, K. (2016). Analyzing social relations for recommending academic conferences. In *HotPOST '16*, *Proceedings of the 8th ACM international workshop on hot topics in planet-scale mobile computing and online social networking* (pp. 37–42). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2944789.2944871.

Beltagy, I., Lo, K., & Cohan, A. (2019). *SciBERT: A pretrained language model for scientific text* (pp. 3615–3620). Hong Kong, China: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/D19-1371, URL: https://www.aclweb.org/anthology/D19-1371.

Bhagavatula, C., Feldman, S., Power, R., & Ammar, W. (2018). Content-based citation recommendation. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (Long papers)* (pp. 238–251). New Orleans, Louisiana: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/N18-1022, URL: https://www.aclweb.org/anthology/N18-1022.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. CoRR abs/2005.14165. URL: https://arxiv.org/abs/2005.14165. arXiv:2005.14165.

Cai, X., Han, J., Pan, S., & Yang, L. (2018). Heterogeneous information network embedding based personalized query-focused astronomy reference paper recommendation. *International Journal of Computational Intelligence Systems*, *11*, 591. http://dx.doi.org/10.2991/ijcis.11.1.44.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. CoRR abs/2005.12872. URL: https://arxiv.org/abs/2005.12872. arXiv:2005.12872.

Chaiwanarom, P., & Lursinsap, C. (2015). Collaborator recommendation in interdisciplinary computer science using degrees of collaborative forces, temporal evolution of research interest, and comparative seniority status. *Knowledge-Based Systems*, *75*, 161–172. http://dx.doi.org/10.1016/j.knosys.2014.11.029.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International conference on learning representations*. URL: https://openreview.net/forum?id=r1xMH1BtvB.

Cohan, A., Ammar, W., van Zuylen, M., & Cady, F. (2019). Structural scaffolds for citation intent classification in scientific publications. In *Proceedings of the 2019 conference of the North American Chapter of the association for computational linguistics: Human language technologies, volume 1 (Long and short papers)* (pp. 3586–3596). Minneapolis, Minnesota: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/N19-1361, URL: https://www.aclweb.org/anthology/N19-1361.

Cramer, J. (2002). *The origins of logistic regression*: *Tinbergen Institute Discussion Papers*, Tinbergen Institute, http://dx.doi.org/10.2139/ssrn.360300.

Cuong, D. V., Nguyen, D. H., Huynh, S., Huynh, P., Gurrin, C., Dao, M.-S., Dang-Nguyen, D.-T., & Nguyen, B. T. (2020). A framework for paper submission recommendation system. In *ICMR '20*, *Proceedings of the 2020 international conference on multimedia retrieval* (pp. 393–396). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3372278.3391929.

Cutler, A., Cutler, D., & Stevens, J. (2011). Random forests. In *Machine learning - ML, Vol. 45* (pp. 157–176). http://dx.doi.org/10.1007/978-1-4419-9326-7_5.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 2978–2988). Florence, Italy: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/P19-1285, URL: https://www.aclweb.org/anthology/P19-1285.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Feng, X., Zhang, H., Ren, Y., Shang, P., Zhu, Y., Liang, Y., Guan, R., & Xu, D. (2019). The deep learning–based recommender system "Pubmender" for choosing a biomedical publication venue: Development and validation study. *Journal of Medical Internet Research*, *21*(5), Article e12957. http://dx.doi.org/10.2196/12957, URL: http://www.jmir.org/2019/5/e12957/.

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis - nonparametric discrimination: Consistency properties. *International Statistical Review*, *57*, 238.

Habib, R., & Afzal, M. T. (2019). Sections-based bibliographic coupling for research paper recommendation. *Scientometrics*, *119*, 643–656.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hong, K., Jeon, H., & Jeon, C. (2013). Personalized research paper recommendation system using keyword extraction based on userprofile. *Journal of Convergence Information Technology*, *8*, 106–116.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87. http://dx.doi.org/10.1162/neco.1991.3.1.79.

Jannach, D., & Jugovac, M. (2019). Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems*, *10*(4), 1–23. http://dx.doi.org/10.1145/3370082.

Jeong, C., Jang, S., Park, E., & Choi, S. (2020). A context-aware citation recommendation model with BERT and graph convolutional networks. *Scientometrics*, *124*(3), 1907–1922. http://dx.doi.org/10.1007/s11192-020-03561-, URL: https://ideas.repec.org/a/spr/scient/v124y2020i3d10.1007_s11192-020-03561-y.html.

Jordan, M. I. (1986). *Serial order: a parallel distributed processing approach*: *Technical report, June 1985-March 1986*, URL: https://www.osti.gov/biblio/6910294.

Kim, J. (2020). Brain activation-based filtering: a scholarly journal recommendation model for human neuroimaging studies. *BioRxiv*, http://dx.doi.org/10.1101/2020.10.06.327684, URL: https://www.biorxiv.org/content/early/2020/10/06/2020.10.06.327684.

Klamma, R., Pham, M. C., & Cao, Y. (2009). You never walk alone: Recommending academic events based on social network analysis. In J. Zhou (Ed.), *Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering*: *vol. 4*, *Complex sciences, first international conference, complex 2009, Shanghai, China, February 23-25, 2009. Revised papers, Part 1* (pp. 657–670). Springer, http://dx.doi.org/10.1007/978-3-642-02466-5_64.

Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 conference on empirical methods in natural language processing: System demonstrations* (pp. 66–71). Brussels, Belgium: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/D18-2012, URL: https://www.aclweb.org/anthology/D18-2012.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A Lite BERT for self-supervised learning of language representations. In *International conference on learning representations*. URL: https://openreview.net/forum?id=H1eA7AEtvS.

Li, Y., Wang, R., Nan, G., Li, D., & Li, M. (2021). A personalized paper recommendation method considering diverse user preferences. *Decision Support Systems*, *146*, Article 113546.

Liu, H., Kou, H., Yan, C., & Qi, L. (2020). Keywords-driven and popularity-aware paper recommendation based on undirected paper citation graph. *Complexity*, *2020*, 1–15. http://dx.doi.org/10.1155/2020/2085638.

Liu, Z., Xie, X., & Chen, L. (2018). Context-aware academic collaborator recommendation. In *KDD '18*, *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1870–1879). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3219819.3220050.

Lu, Y., He, Y., Cai, Y., Peng, Z., & Tang, Y. (2021). Time-aware neural collaborative filtering with multi-dimensional features on academic paper recommendation. In *2021 IEEE 24th international conference on computer supported cooperative work in design (CSCWD)* (pp. 1052–1057). http://dx.doi.org/10.1109/CSCWD49262.2021.9437673.

Luong, H., Huynh, T., Gauch, S., Do, L., & Hoang, K. (2012). Publication venue recommendation using author network's publication history. In J.-S. Pan, S.-M. Chen, & N. T. Nguyen (Eds.), *Intelligent information and database systems* (pp. 426–435). Berlin, Heidelberg: Springer Berlin Heidelberg.

Marius, P., Balas, V., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, *8*.

Medvet, E., Bartoli, A., & Piccinin, G. (2014). Publication venue recommendation based on paper abstract. In *2014 IEEE 26th international conference on tools with artificial intelligence* (pp. 1004–1010). http://dx.doi.org/10.1109/ICTAI.2014.152.

Nguyen, D., Huynh, S., Huynh, P., Dinh, C. V., & Nguyen, B. T. (2021). S2CFT: A new approach for paper submission recommendation. In T. Bureš, R. Dondi, J. Gamper, G. Guerrini, T. Jurdziński, C. Pahl, F. Sikora, & P. W. Wong (Eds.), *SOFSEM 2021: Theory and practice of computer science* (pp. 563–573). Cham: Springer International Publishing.

Peiris, D., & Weerasinghe, R. (2015). Citation network based framework for ranking academic publications and venues. In *2015 fifteenth international conference on advances in ICT for emerging regions (ICTer)* (pp. 146–151). http://dx.doi.org/10.1109/ICTER.2015.7377681.

Pradhan, T., Gupta, A., & Pal, S. (2020). HASVRec: A modularized hierarchical attention-based scholarly venue recommender system. *Knowledge-Based Systems*, *204*, Article 106181. http://dx.doi.org/10.1016/j.knosys.2020.106181, URL: http://www.sciencedirect.com/science/article/pii/S0950705120304135.

Pradhan, T., & Pal, S. (2019). A hybrid personalized scholarly venue recommender system integrating social network analysis and contextual similarity. *Future Generation Computer Systems*, http://dx.doi.org/10.1016/j.future.2019.11.017.

Pradhan, T., & Pal, S. (2020). CNAVER: A Content and network-based academic venue recommender system. *Knowledge-Based Systems*, *189*, Article 105092. http://dx.doi.org/10.1016/j.knosys.2019.105092, URL: http://www.sciencedirect.com/science/article/pii/S0950705119304691.

Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. URL: https://openai.com/blog/better-language-models/.

Rumelhart, D., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation.

Safa, R., Mirroshandel, S., Javadi, S., & Azizi, M. (2018). Venue recommendation based on paper's title and co-authors network. *Journal of Informaton Systems and Telecommunication*, *6*, http://dx.doi.org/10.7508/jist.2018.21.005.

Sakib, N., Ahmad, R. B., Ahsan, M., Based, M. A., Haruna, K., Haider, J., & Gurusamy, S. (2021). A hybrid personalized scientific paper recommendation approach integrating public contextual metadata. *IEEE Access*, *9*, 83080–83091. http://dx.doi.org/10.1109/ACCESS.2021.3086964.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long papers)* (pp. 1715–1725). Berlin, Germany: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/P16-1162, URL: https://www.aclweb.org/anthology/P16-1162.

Son, H., Phong, H., Dac, N., Cuong, D. V., & Binh, N. T. (2020). S2RSCS: An efficient scientific submission recommendation system for computer science. In *The 33th international conference on industrial, engineering & other applications of applied intelligent systems*.

Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation. ArXiv, abs/1908.08962.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *NIPS*.

Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. *Knowledge-Based Systems*, *157*, 1–9. http://dx.doi.org/10.1016/j.knosys.2018.05.001, URL: http://www.sciencedirect.com/science/article/pii/S0950705118302107.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., ... Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144. URL: http://arxiv.org/abs/1609.08144. arXiv:1609.08144.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Yang, F., Yang, H., Fu, J., Lu, H., & Guo, B. (2020). Learning texture transformer network for image super-resolution. CoRR abs/2006.04139. URL: https://arxiv.org/abs/2006.04139. arXiv:2006.04139.

Ye, L., Rochan, M., Liu, Z., & Wang, Y. (2019). Cross-modal self-attention network for referring image segmentation. CoRR abs/1904.04745. URL: http://arxiv.org/abs/1904.04745. arXiv:1904.04745.

Zhao, W., Wu, R., Dai, W., & Dai, Y. (2015). Research paper recommendation based on the knowledge gap. In *2015 IEEE international conference on data mining workshop (ICDMW)* (pp. 373–380). http://dx.doi.org/10.1109/ICDMW.2015.40.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). Deformable DETR: Deformable transformers for end-to-end object detection. CoRR abs/2010.04159. URL: https://arxiv.org/abs/2010.04159. arXiv:2010.04159.