

Received July 2, 2021, accepted July 21, 2021, date of publication August 3, 2021, date of current version August 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3102005

Toward Building an Academic Search Engine Understanding the Purposes of the Matched Sentences in an Abstract

LI-YUAN HSU¹, CHIA-HAO KAO², I-SHENG JHENG³, AND HUNG-HSUAN CHEN¹

¹Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA

²Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan

³Department of Computer Science and Information Engineering, National Central University, Taoyuan 32001, Taiwan

Corresponding author: Hung-Hsuan Chen (hhchen@g.ncu.edu.tw)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant 107-2221-E-008-077-MY3.

ABSTRACT This paper introduces an automatic approach to understand the purposes of each sentence in the abstract of an academic document. Specifically, computers can label each sentence in the abstract as being related to one or several of six aspects – “BACKGROUND”, “OBJECTIVES”, “METHODS”, “RESULTS”, “CONCLUSIONS”, and “OTHERS”. Experimental results obtained on a real dataset show that the labeling methodology outperforms baseline methods. We also build a prototype academic search engine to demonstrate the use of this new design. Users may search for articles containing keywords related to any of these six aspects to better meet their search goals.

INDEX TERMS Bidirectional LSTM, hierarchical LSTM, document understanding, specialty search engine.


I. INTRODUCTION

With the popularity of academic search engines and bibliographical databases, researchers have become largely dependent on these tools for literature searches and surveys [12], [25], [34]. These search tools efficiently discover and rank articles relevant to a querier’s search keywords based on text indexing and various ranking factors, e.g., TF-IDF, citation count, and published date [2], [4], [31]. However, a search index is essentially equivalent to a lookup table. In other words, these tools simply search for documents containing the query term without knowing the querier’s intention and the document authors’ reasons for using this term. Consequently, the querier has to read the titles, abstracts, sentences near the matched term, and perhaps other sections in the returned list of documents to determine the fitness of each document relative to their search intention.

As an example, we searched for the term “KNN classifier” on Google Scholar. At the time of writing this paper, the top 5 results included two papers applying the k -nearest neighbors algorithm on certain application domains [20], [21], two papers discussing strategies to improve the k -nearest neighbors algorithm [11], [27], and an empirical study that compares the naïve Bayesian classifier with the

k -nearest neighbors classifier [15]. If a querier is interested in investigating the details of the KNN algorithm, perhaps the two papers proposing improvement strategies ([27] and [11]) will better serve the querier’s needs. On the other hand, if the querier wants to know the use cases of the KNN, the two application papers ([20] and [21]) are likely better choices. Consequently, one universal ranking list for different queriers with different purposes is not enough. We should allow queriers to specify their “purposes” and then rank the documents accordingly, or perhaps the academic search engines should list not only the snippets of the matched sentences but also the purposes of these sentences so that the queriers can efficiently judge the fitness between the returned documents and their querying purposes.

Motivated by the above scenario, our first objective of the paper is to propose a new model to automatically label the purposes of sentences in the abstract of an academic document. We surmise that each sentence in the abstract of an academic document may express certain characteristics related to one of the following six aspects: “BACKGROUND”, “OBJECTIVES”, “METHODS”, “RESULTS”, “CONCLUSIONS”, and “OTHERS”. We design a machine learning algorithm to tag each sentence with one or a few of these six aspects automatically. Experimental results on a real dataset show that the Samples-F1 score reaches 0.71, suggesting that our model can successfully

The associate editor coordinating the review of this manuscript and approving it for publication was Fu Lee Wang .

label the purposes of a large portion of the sentences in the abstract.

Our second objective is to prototype an academic search engine that understands the purposes of the matched sentences. The prototyped system contains 7,000 academic documents collected from arXiv.org. Each abstract sentence is automatically labeled by our proposed algorithm. The system indexes all the sentences along with their corresponding purpose labels. A querier may choose to search for sentences of certain aspects that better serve her/his intention. Additionally, we show each sentence in the abstract along with its purpose label(s) in the search result page so that readers may review the structure of the sentences more easily.

The rest of the paper is organized as follows. In Section II, we review various academic search engines, including metadata-based academic search engines, crawling-based academic search engines, and specialty academic search engines. We also review previous works on automatic labeling of abstract sentences. Section III introduces the methodology used to predict the sentence labels. Section IV shows the experimental settings and the results of our model and several baseline models. We present a prototype system to demonstrate the intention-based search system in Section V. Finally, we conclude the work and discuss future work in Section VI.

II. RELATED WORK

Academic search engines and bibliographical databases have become essential tools for literature surveys in academia. This section reviews two categories of academic search engines — metadata-based academic search engines and crawling-based academic search engines. Additionally, we discuss certain academic search engines that provide unique functionalities. Finally, we review previous studies on abstract sentence classification.

A. METADATA-BASED ACADEMIC SEARCH ENGINES

Metadata-based academic search engines refer to search engines that compile the metadata primarily from humans. Since metadata collection, editing, and maintenance are laborious tasks, metadata-based digital libraries are usually related to the publishers, as they have the resources and motivation for manual editing.

In the computing and information fields, the ACM and IEEE are probably the two most important academic organizations. These two organizations each have their own digital libraries, namely, the ACM Digital Library¹ and IEEE Xplore.² Since these two organizations can usually obtain metadata directly from the organizers of academic conferences or from journal editors, the metadata are usually precise and credible. Another popular service, the bibliography reference DBLP,³ contains information on the titles, authors,

conference or journal names, publication years, and several other pieces of bibliographical information for more than 5 million academic documents, mostly in the domains of computer science and information science [19]. The metadata of DBLP undergoes a strict editing and cleaning process and thus is usually credible. However, DBLP provides only metadata search and not full-text search.⁴ While the metadata of the ACM digital library, IEEE Xplore, and DBLP are highly accurate, they require laborious manual editing; thus, the maintenance cost is very high.

B. CRAWLING-BASED ACADEMIC SEARCH ENGINES

Crawling-based academic search engines collect scientific documents from the Internet and extract the contents (e.g., title, author(s), abstract, references, and publication date) from the documents based on various artificial-intelligence-related technologies, such as natural language processing, information extraction, information retrieval, optical character recognition (OCR) techniques, and many more [1], [32]. Representative crawling-based academic search engines include Google Scholar [13], CiteSeerX [31], and Microsoft Academic Search [26].

Since this type of system is highly autonomous and requires little manual intervention during the metadata extraction process, it might be easier to scale. For example, Khabsa and Giles estimated that Google Scholar owned 88% (100 million out of 114 million) of the English scholarly documents that were available on the Internet in 2014 [17]. However, because the extracted information could be less precise, minimal corrections or editing might still be needed [3], and some of these systems, e.g., Google Scholar, allow authorized users (e.g., registered users or document authors) to update the generated metadata. Some of these platforms provide only an abstract search [5] because of various limitations, e.g., copyright infringement issues.

C. SPECIALTY ACADEMIC SEARCH ENGINES

Specialty academic search engines provide unique search services for special needs. For example, CollabSeer suggests potential academic collaborators of specified domains within a querier's academic social circle [2]. CSSeer, AMiner, and Microsoft Academic recommend experts of the target area [4], [26], [28]. RefSeer recommends references based on the given abstract text [14]. TableSeer extracts and indexes the table captions and provides a unique table search functionality [22]. AlgorithmSeer identifies and indexes the pseudocodes and the captions of the algorithms in academic documents [29].

While various academic search engines and bibliographical databases are available, to the best of our knowledge, there are no platforms that integrate sentences and their corresponding computer-generated purpose tags into the new functionality of “sentence intention search”.

¹<https://dl.acm.org/>

²<https://ieeexplore.ieee.org/>

³<https://dblp.org/>

⁴ <https://dblp.org/faq/16154928.htmlPage>

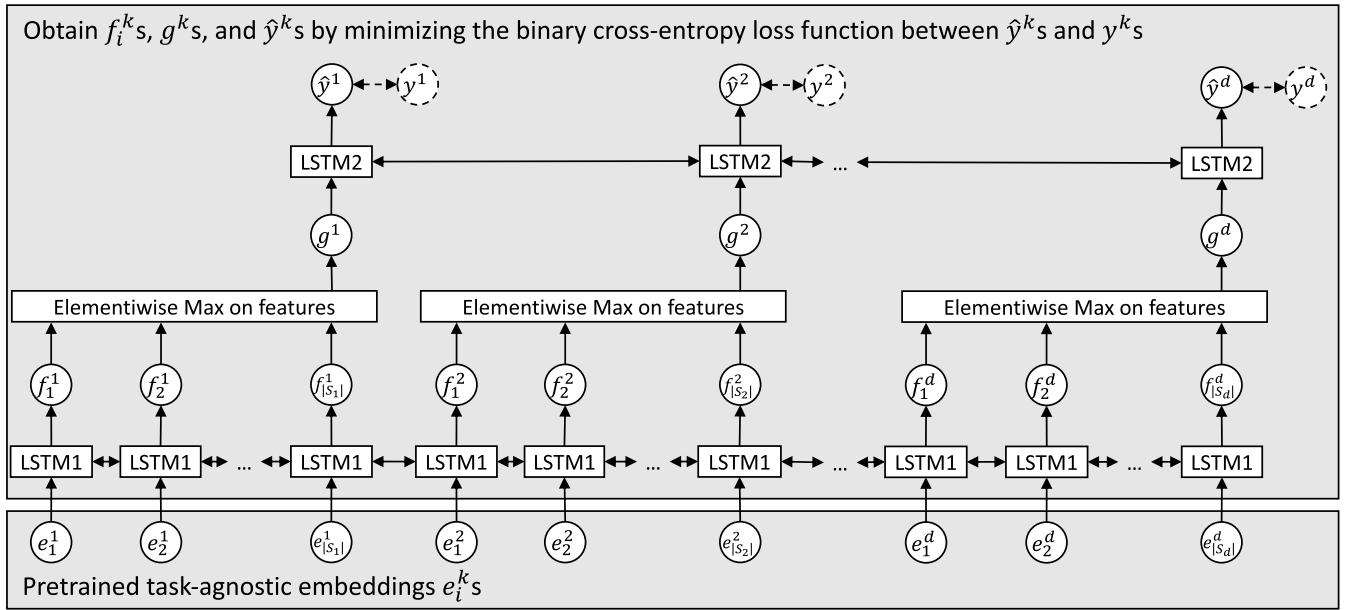


FIGURE 1. The sentence purpose identification Hierarchical LSTM-W2V model. The task-agnostic embeddings are obtained by applying CBOW on the sentences in all abstracts. The predicted labels \hat{y}^k s are predicted by integrating the sentence embeddings g^k s with a bidirectional LSTM layer. The sentence embeddings g^k s are generated by integrating the task-dependent embeddings f_i^k s based on another bidirectional LSTM layer.

D. ABSTRACT SENTENCE CLASSIFICATION

Some previous works aim to classify abstract sentences into proper headings [6]. Early works in the area require manually defining sentence features as the input for supervised classifiers. For example, SeCBLiS [33] utilizes the sentence position, the presence of an auxiliary verb in a sentence, and some other sentence features as the input of a support vector machine.

Recent works mainly apply deep neural networks (DNNs) for this task. Since certain types of DNNs (e.g., variants of recurrent neural networks) allow us to use features with variable lengths; we may use the word tokens or the features generated from word tokens (e.g., word embeddings) as the input even when each sentence has different numbers of word tokens. We list two works whose designs are similar to our design and point out the differences. First, bi-ANN [6], [7] utilizes word embeddings as the input of an LSTM model to generate sentence embeddings. To obtain the relationship among different sentences, bi-ANN integrates the conditional probability of the class of the current sentence given the class of the previous sentence. However, such a design assumes that the class of the current sentence is conditionally independent of the classes of the future sentences and the previous n ($n \geq 2$) sentences given the class of the previous sentence. The other model, Word-BiGRU [10] employs word embeddings within the same sentence to generate sentence embeddings, which are further used to label the class of the sentence. The Word-BiGRU model utilizes convolution layers with filter sizes of 5 to integrate the words within a sentence. However, such a setting enforces each word to influence only the previous two words and the following two words.

Our model is different from the above models mainly in that we consider not only all words in a sentence to generate sentence embeddings but also the relationship among all sentence labels in an abstract. Additionally, previous methods model the problem as a multiclass classification problem, which presumes that each sentence belongs to precisely one class. Instead, we model the task as a multilabel classification problem so that each sentence may belong to multiple classes. Although this is a trivial issue that can be easily solved by changing the shape and the loss function of the output layer, none of the abovementioned models did this.

III. SENTENCE PURPOSE PREDICTION MODEL

This section introduces the methodology used to predict the purposes of a sentence. Each sentence may contain more than one purpose; thus, we model the problem as a multilabel classification problem, i.e., each instance may belong to one or multiple purpose classes. We provide a sample abstract and the purpose labels of the sentences in Table 1.

The entire prediction model consists of two parts: task-agnostic word embedding generation and sentence purpose label prediction. Figure 1 shows an overview of the entire model.

A. TASK-AGNOSTIC WORD EMBEDDING GENERATION

To generate the task-agnostic word embeddings, we segment the abstract text into sentences $S^1, S^2, \dots, S^k, \dots, S^D$ (assuming D total sentences in all abstracts). Each sentence S^k consists of words $t_1^k, \dots, t_{|S_k|}^k$, where $|S_k|$ is the word count of the sentence S^k . Words within the same sentence are fed into the standard continuous bag-of-words (CBOW) model [23] (with the negative sampling strategy) to generate the

TABLE 1. A sample abstract and the corresponding sentence labels. The original paper can be downloaded from <https://arxiv.org/abs/1110.1930>.

Label	Sentence
BACKGROUND	Low-density parity-check (LDPC) codes on symmetric memoryless channels have been analyzed using statistical physics by several authors.
OBJECTIVES	In this paper, statistical mechanical analysis of LDPC codes is performed for asymmetric memoryless channels and general Markov channels.
RESULTS, CONCLUSIONS	It is shown that the saddle point equations of the replica symmetric solution for a Markov channel is equivalent to the density evolution of the belief propagation on the factor graph representing LDPC codes on the Markov channel.
METHODS	The derivation uses the method of types for Markov chain.

word embeddings $e_1^k, \dots, e_{|S_k|}^k$ of the words in sentence S^k . We call these embeddings “task-agnostic embeddings”, as these embeddings are trained in a self-supervised fashion, i.e., they are independent of the downstream task. Consequently, they are irrelevant to the target labels. Essentially, the CBOW model predicts the current word based on the neighboring context words within the specified window size. Equation 1 shows the objective function. The optimizer finds the word embeddings e_j^k to maximize the objective function J .

$$J(\mathbf{e}) = \log \left(\sum_{k=1}^D \sum_{i=1}^{|S_k|} \sum_{-m \leq j \leq m, m \neq 0} P(e_i^k | e_{i+j}^k) \right), \quad (1)$$

where D is the total number of sentences in all abstracts, m is the user-specified window size, $\mathbf{e} = [e_i^k] (i = 1, \dots, |S_k|, k = 1, \dots, D)$, and $P(e_i^k | e_{i+j}^k)$ is the probability of observing the target word t_i^k (with embedding e_i^k) given the contextual word t_{i+j}^k (with embedding e_{i+j}^k). The conditional probability is defined by Equation 2.

$$P(e_i^k | e_{i+j}^k) = \log \sigma(e_i^k \cdot e_{i+j}^k) + \sum_{\ell \sim p(w)} \left[\log \sigma(-e_\ell \cdot e_{i+j}^k) \right], \quad (2)$$

where \cdot is the inner product operator, $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, and $p(w) \propto f(w)^{0.75}$ is the negative sampling function ($f(w)$ is the empirical frequency distribution of word w).

B. PREDICTING SENTENCE PURPOSE LABELS BASED ON HIERARCHICAL LSTM

The sentence label prediction module contains three parts: task-dependent word embedding generation, sentence embedding generation, and finally sentence label prediction. The task-dependent word embedding and sentence embedding together form a Hierarchical LSTM. This structure may look similar to a multilayer LSTM. However, it is different, because in multilayer LSTM, the output of an LSTM layer is the input of the next LSTM layer, whereas in Hierarchical LSTM, the output of an LSTM layer is integrated before being sent to the next LSTM layer. We explain the three parts in the following paragraphs.

First, we feed the task-agnostic word embeddings e_j^k s into a bidirectional LSTM layer. The outputs f_j^k s of the LSTM layer

are called the task-dependent word embeddings because these word embeddings are influenced by the target labels. Specifically, the task-dependent word embeddings are computed by Equation 3:

$$f_j^k = o \odot \tanh(c_{j-1}), \quad (3)$$

where \odot is the elementwise multiplication operator and o and c_{j-1} are the output gate and the cell state, which are defined in Equation 4 and Equation 5, respectively.

$$o = \sigma \left(\mathbf{w}_o^T \begin{bmatrix} h_{j-1} \\ e_j^k \end{bmatrix} \right), \quad (4)$$

where \mathbf{w}_o is a vector of the parameters to learn.

$$c_{j-1} = f \odot c_{j-2} + i \odot \tilde{c}, \quad (5)$$

where f , i , and \tilde{c} are the forget gate, input gate, and candidate cell state, which are computed based on Equation 6, Equation 7, and Equation 8, respectively.

$$f = \sigma \left(\mathbf{w}_f^T \begin{bmatrix} h_{j-1} \\ e_j^k \end{bmatrix} \right), \quad (6)$$

where \mathbf{w}_f is a vector of the parameters to learn.

$$i = \sigma \left(\mathbf{w}_i^T \begin{bmatrix} h_{j-1} \\ e_j^k \end{bmatrix} \right), \quad (7)$$

where \mathbf{w}_i is a vector of the parameters to learn.

$$\tilde{c} = \tanh \left(\mathbf{w}_c^T \begin{bmatrix} h_{j-1} \\ e_j^k \end{bmatrix} \right), \quad (8)$$

where \mathbf{w}_g is a vector of the parameters to learn.

Next, we compute the elementwise maximum on task-dependent word embeddings of the same sentence (i.e., $f_1^k, f_2^k, \dots, f_{|S_k|}^k$) to obtain the sentence embedding g^k for a sentence S_k . Specifically, if we use $g^k[\ell]$ to denote the ℓ th element in g^k , then $g^k[\ell]$ is computed by Equation 9.

$$g^k[\ell] = \max(f_1^k[\ell], \dots, f_{|S_k|}^k[\ell]), \quad (9)$$

where $f_m^k[\ell]$ denotes the ℓ th element of the embedding from the m th word in sentence S_k .

Finally, the sentence embeddings g^k s are fed into another bidirectional LSTM layer that outputs \hat{y}^k to predict the multi-hot-encoded output label y^k . Multi-hot encoding is used because a sentence may belong to multiple classes. We use Adam (with a learning rate of 0.001) as the optimizer [18].

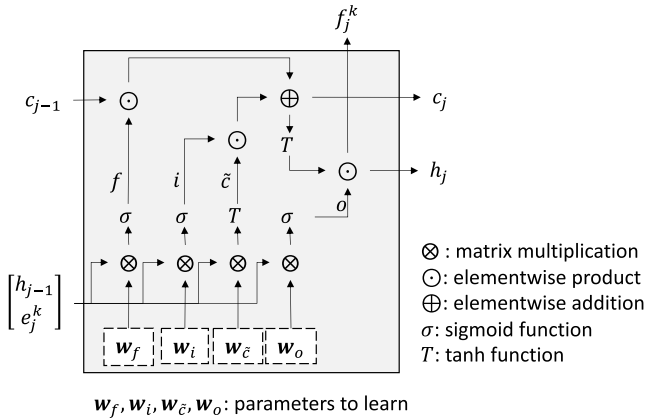


FIGURE 2. The computational graph used to generate task-dependent embedding f_j^k from the task-agnostic embeddings. This component corresponds to one LSTM cell in Figure 1. The variables $f, i, o,$ and \tilde{c} correspond to the forget gate, input gate, output gate, and candidate cell state. The variable c_j represents the cell state at the j th word. Consequently, the long-term information is memorized by the cell states (i.e., c_j s), and the short-term information is preserved by h_j s.

We use the binary cross-entropy as the loss function, which is defined by Equation 10

$$L(\hat{y}, y) = \sum_{i=1}^d \sum_{\ell=1}^m \left[-y^i[\ell] \log \hat{y}^i[\ell] - (1 - y^i[\ell]) \log (1 - \hat{y}^i[\ell]) \right], \tag{10}$$

where $\hat{y} = [\hat{y}^1, \dots, \hat{y}^d]^T, y = [y^1, \dots, y^d]^T,$ and $y^i[\ell]$ and $\hat{y}^i[\ell]$ represent the ℓ th element of y^i and \hat{y}^i , respectively ($y^k[\ell] \in \{0, 1\}, 0 \leq \hat{y}^k[\ell] \leq 1$).

The entire training process is shown in Figure 1. The task-agnostic word embeddings e_j^k are obtained through a pretraining process, which was introduced in Section III-A. The other parameters ($w_f, w_i, w_g,$ and w_o) are learned by minimizing the binary cross-entropy loss shown in Equation 10.

Because there are many variables involved in the second part (the prediction of sentence purpose labels), we show the computation graph used to obtain f_j^k from e_j^k in Figure 2 for better illustration. Figure 2 corresponds to one LSTM1 cell in Figure 1. For the LSTM2 cell, one needs to replace only the input and output from e_j^k and f_j^k with g^k and \hat{y}^k , respectively. Note that the parameters ($w_f, w_i, w_g,$ and w_o) for LSTM1 and LSTM2 are different.

IV. EXPERIMENTS ON PREDICTING SENTENCE PURPOSE LABELS

This section introduces the experimental dataset, setup, and compares the experimental results of predicting sentence purpose labels.

A. EXPERIMENTAL DATASET

Our experimental dataset contains 7,000 academic documents collected from the arXiv.org e-print archive. Each

TABLE 2. The distribution of the purpose labels. The “OTHERS” label accounts for less than 2% of all labels.

Label Name	Counts	Percentage (%)
BACKGROUND	13,353	24.58
OBJECTIVES	9,329	17.17
METHODS	13,655	25.14
RESULTS	11,772	21.67
CONCLUSIONS	5,313	9.78
OTHERS	901	1.66
Sum	54,323	100

TABLE 3. The distribution of the number of labels per sentence. While most sentences have 1 or 2 labels, few sentences may have 3, 4, or even up to 5 labels.

Number of Labels per Sentence	Counts	Percentage (%)
1	40,179	85.73
2	6,059	12.93
3	527	1.12
4	65	0.14
5	37	0.08

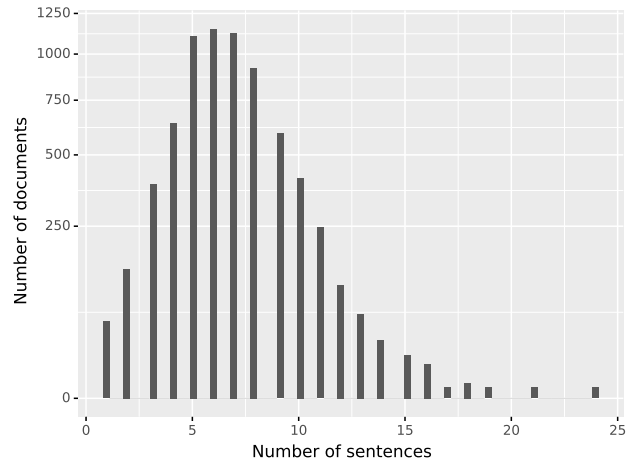


FIGURE 3. Number of sentences per document (the y axis is on the square-root scale). While most documents have four to nine sentences in their abstracts, few documents have very short or very long abstracts.

document includes the title, author name(s), abstract, created date, and paper subject, e.g., networking and internet architecture (cs.NI) and computer vision and pattern recognition (cs.CV). Each sentence in the abstracts of these documents is labeled with one or several of the six aspects (BACKGROUND, OBJECTIVES, METHODS, RESULTS, CONCLUSIONS, AND OTHERS). For a sample abstract, the label of each sentence is shown in Table 1. Table 2 shows the appearance counts and the corresponding percentages of the six labels. Because each sentence may have multiple labels, we show the distribution of the number of labels each sentence has in Table 3.

These 7,000 documents contains 46,867 sentences in total, i.e., each document has 6.70 sentences on average. Figure 3 shows the histogram of the number of sentences per document. The y-axis is in the square-root scale.

We divide the dataset into a training dataset (including 6, 300 articles) and a testing dataset (including 700 articles) to test the performance of our model. We further divide the training dataset into training and validation sets if the hyperparameters need to be fine-tuned.

B. EXPERIMENTAL METRICS

We use confusion matrices and the precision, recall, and F1 score to demonstrate the effectiveness of the model. However, because we are dealing with a multilabel classification problem, some of these evaluation metrics require adjustment, as described below.

First, we cannot directly use a large confusion matrix to represent the relationship between the predicted and the real classes because a general confusion matrix allocates each instance into one cell based on the predicted and actual class of the instance, while in our scenario, sentences (instances) may have multiple labels. Consequently, we use the one-vs-rest approach to produce six confusion matrices; i.e., each confusion matrix represents the result of treating one class as positive and the other classes as negative.

Second, we show the precision, recall, and F1 score for each class, again using the one-vs-rest approach. Once we obtain the precision, recall, and F1 score for each class, we compute the weighted precision/recall/F1 score to represent the overall performance. The weighted precision/recall/F1 score can be regarded as an extension of the macro-precision/recall/F1 score because the weighted version utilizes the support of each class (i.e., the correct number of instances for each class) to determine the relative weights. Equation 11 shows the weighted metrics.

$$\text{weighted } \langle X \rangle = \sum_{i=1}^C w_i \langle X_i \rangle, \quad (11)$$

where w_i is the support of class i dividing the number of instances, $\langle X \rangle$ denotes a metric score (i.e., the precision, recall, or F1 score in this paper), and $\langle X_i \rangle$ is the metric score for class i .

We also report the micro-precision/recall/F1 score. Let TP_i , FP_i , and FN_i denote the numbers of true positives, false positives, and false negatives, respectively, for class i . The micro-precision/recall/F1 score are expressed as Equations 12, 13, and 14, respectively.

$$\text{micro-precision} = \frac{\sum_{i=1}^6 TP_i}{\sum_{i=1}^6 (TP_i + FP_i)} \quad (12)$$

$$\text{micro-recall} = \frac{\sum_{i=1}^6 TP_i}{\sum_{i=1}^6 (TP_i + FN_i)} \quad (13)$$

$$\text{micro-F1} = \frac{2 \times \text{micro-precision} \times \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}} \quad (14)$$

Although the weighted and micro-metrics assign proper weights to each class, studies suggest that the samples-precision/recall/F1 score are probably more appropriate for

TABLE 4. An example of the Position model for documents with 6 sentences. Consequently, when given an abstract with 6 sentences, the Position model always predicts the labels of the six sentences as BACKGROUND, BACKGROUND, METHODS, METHODS, RESULTS, and RESULTS.

Sentence position	Most frequent label
1	BACKGROUND
2	BACKGROUND
3	METHODS
4	METHODS
5	RESULTS
6	RESULTS

multilabel classification problems [9], [16]. The argument is that the weighted and the micro-metrics may be highly affected by the instances with many labels and that the sample metrics treat each instance equally, as the sample metrics first compute the precision, recall, and F1 score for each instance individually and then compute the average across all instances. We report the weighted average, micro-average, and sample average, as each may have its own advantages and disadvantages.

C. BASELINE MODELS

We compare our model with eight baseline methods. The first two models are naïve methods based on simple heuristics. The third, fourth, and fifth models are previous methods that were used for the same task. Finally, the sixth, seventh, and eighth models are variations of our final Hierarchical LSTM-W2V model. The performance of the last three baselines, along with our final Hierarchical LSTM-W2V model, may help determine the reasons why the Hierarchical LSTM-W2V model works.

The first model is a naïve baseline, which computes the most frequent label ℓ_i in the training data and always predicts a sentence as ℓ_i . We call this simple baseline the “Majority model”, as this model always returns the majority label in the training dataset.

The second baseline model, which we call the “Position model”, computes the label of a sentence based on two factors – the number of sentences in the abstract and the position of the sentence we want to label. During training, we categorize a paper into group $g^{(i)}$ if this paper has i sentences in the abstract. We further define $g_j^{(i)}$ as the j th paper in group $g^{(i)}$, $g_{j,k}^{(i)}$ as the k th sentence in $g_j^{(i)}$, and $\ell_{j,k}^{(i)}$ as the label(s) of $g_{j,k}^{(i)}$. During prediction, when given a paper with m sentences, we predict its n th sentence by the majority label in $[\ell_{1,n}^{(m)}, \ell_{2,n}^{(m)}, \dots, \ell_{|g^{(m)}|,n}^{(m)}]$ (assuming there are $|g^{(m)}|$ papers belonging to group $g^{(m)}$ in the training data). For instance, Table 4 shows the most frequent label for each sentence position in group $g^{(6)}$. Consequently, when given a new paper with 6 sentences in the abstract, the sentence labels are predicted as “BACKGROUND”, “BACKGROUND”, “METHODS”, “METHODS”, “RESULTS”, and “RESULTS”, respectively, when using the Position model.

The third model is SeCBLiS [33]. This model defines sentence features in an ad hoc manner. The sentence features

TABLE 5. A comparison of the micro-precision/recall/f1 score, weighted-precision/recall/F1 score, and samples-precision/recall/F1 score of different methods. We highlight the highest score in each column in bold face and the second highest score in each column with an underscore.

	Micro			Weighted			Samples		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Majority	0.28	0.25	0.26	0.07	0.25	0.11	0.28	0.27	0.27
Position	0.56	0.48	0.51	0.54	0.48	0.45	0.56	0.50	0.52
SeCBLiS [33]	0.61	0.48	0.53	0.60	0.48	0.52	0.59	0.51	0.54
bi-ANN [6], [7]	0.65	0.60	0.62	0.62	0.60	0.60	0.61	0.63	0.61
Word-BiGRU [10]	0.65	0.63	<u>0.64</u>	0.61	0.63	0.62	0.62	0.65	<u>0.64</u>
RF-TFIDF	0.62	0.35	0.44	0.60	0.35	0.42	0.40	0.36	0.37
LSTM-TFIDF	0.62	<u>0.67</u>	<u>0.64</u>	0.60	<u>0.67</u>	<u>0.63</u>	<u>0.63</u>	<u>0.69</u>	<u>0.64</u>
RF-W2V	0.71	0.30	0.42	0.70	0.30	0.39	0.34	0.31	0.32
Hierarchical LSTM-W2V (our model)	<u>0.66</u>	0.74	0.70	<u>0.66</u>	0.74	0.69	0.70	0.77	0.71

include, for example, the sentence position, frequency of selected terms, χ^2 values of selected terms, and average tf-idf score of the terms in the sentence. SeCBLiS uses a support vector machine classifier with a linear kernel to predict the class of a sentence. As the selected terms to compute the frequencies and the χ^2 scores are not reported in [33], we used the position features and the average tf-idf scores as the features in our experiment.

The fourth model is bi-ANN [6], [7]. This model generates each sentence embedding by integrating the word embeddings within a sentence. The model further decides the label of a sentence based on both the current sentence embedding and the label of the previous sentence. However, such a design presumes that the Markov property holds; i.e., the label of a sentence is conditionally independent of the labels of the following sentences and the labels of the previous n sentences ($n \geq 2$) given the previous sentence. Such an assumption is over naïve.

The fifth model, Word-BiGRU [10], also generates each sentence embedding by integrating the word embeddings within each sentence. This model incorporates the relationship among different sentences by a bidirectional gated recurrent unit (GRU). However, Word-BiGRU produces a sentence embedding by applying a convolution layer with a filter size of 5. As a result, each word is related to only the preceding two words and the consecutive two words.

For the sixth baseline model, we regard each sentence as one document to compute a matrix of the TF-IDF features for each word. We build a random forest model that takes the TF-IDF feature vector of a sentence as the input feature to predict the purpose label(s) of this sentence. We call this baseline the “RF-TFIDF model”, as the model uses the random forest model and the TF-IDF feature vectors of the words in the sentences.

The seventh baseline model also uses the TF-IDF features as in the previous model. However, we use a bidirectional LSTM model to predict the label(s) of each sentence. We call this baseline method the LSTM-TFIDF model, as it uses the bidirectional LSTM model and the TF-IDF feature vectors as the input.

In the eighth model, we generate the task-agnostic embeddings for each word, as introduced in Section III-A. For the embeddings of the words in sentence S_j (i.e., $e_1^j, \dots, e_{|S_k|}^j$), we generate the sentence feature X_j for sentence S_j by

computing the elementwise average on $e_1^j, \dots, e_{|S_k|}^j$. In other words, $X_j[p] = (\sum_i e_i^j[p]) / |S_k|$, where $X_j[p]$ and $e_i^j[p]$ denote the p th element of vector X_j and vector e_i^j , respectively. We use X_j as the input features of the random forest model to predict the label(s) of the sentence S_j . We call this baseline method the RF-W2V model, as the model utilizes the word2vec embeddings as the input of the random forest model.

D. RESULTS

1) COMPARISON WITH BASELINE METHODS

Table 5 shows various precision/recall/F1 scores, including the micro-precision/recall/F1 score, weighted-precision/recall/F1 score, and samples-precision/recall/F1 score of our model and the eight baseline models. We highlight the first and the second place of each column by using bold face and an underscore, respectively.

Because we are dealing with a multilabel classification problem with imbalanced target labels (as displayed in Table 2), it is not straightforward to know what a decent F1 score is. In this situation, the F1 score of the Majority model (first row in Table 5) may give us a hint, since always predicting the majority label can lead to the highest expected value of the F1 score being obtained if we do not have any information on the relationship between a sentence and its labels.

The Position model, despite using the position of a sentence as the sole clue to predict the corresponding label(s), yields good predictions. In fact, if we simply compare the various F1 scores, the Position model outperforms the more complicated RF-TFIDF and RF-W2V models, probably because the RF-TFIDF model and the RF-W2V model do not consider the positional information. This result suggests that sentence position is an important factor of determining the purpose labels of a sentence, probably even more important than the word choices.

The three previously proposed models, SeCBLiS, bi-ANN, and Word-BiGRU, perform better than the Position model, probably because these models predict the sentence labels based on the sentence features generated either by manually defined features or by integrating the word embeddings. Consequently, these models may discover the relationship

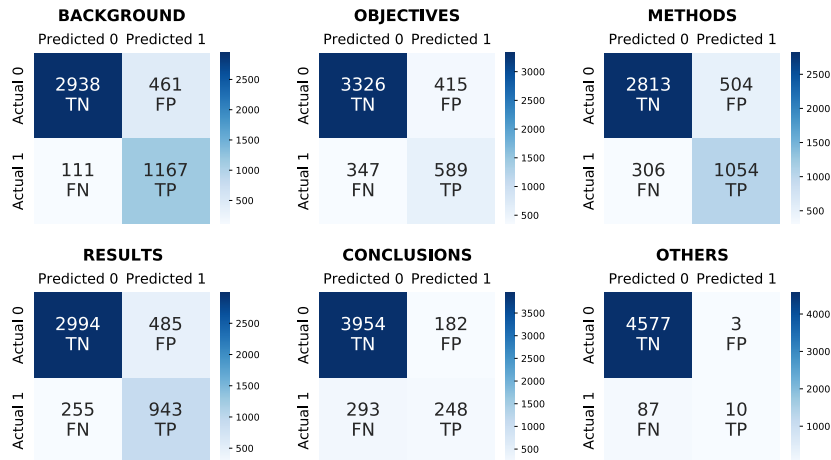


FIGURE 4. The confusion matrices of the six labels when using the Hierarchical LSTM-W2V model. Since we are dealing with a multi-label classification task, we cannot use a large 6×6 confusion matrix to denote the result. Instead, we use six 2×2 matrices, each of which regards one class as positive and the others as negative class.

TABLE 6. The precision, recall, F1 score, and support of each label when using our Hierarchical LSTM-W2V model.

	Precision	Recall	F1 Score	Support
BACKGROUND	0.72	0.91	0.80	1278
OBJECTIVES	0.59	0.63	0.61	936
METHODS	0.68	0.78	0.72	1360
RESULTS	0.66	0.79	0.72	1198
CONCLUSIONS	0.58	0.46	0.51	541
OTHERS	0.77	0.10	0.18	97
Micro Average	0.66	0.74	0.70	5410
Weighted Average	0.66	0.74	0.69	5410
Samples Average	0.70	0.77	0.71	5410

between the word choices and the purposes of the sentences. However, the SeCBLiS model defines sentence features in an ad hoc manner and does not consider the relationship among different sentences. The bi-ANN model generates sentence features systematically (specifically, generating sentence embeddings from word embeddings), but the model considers only the label relationship between two consecutive sentences and ignores all other labels. Finally, the Word-BiGRU model examines the embedding of the target sentence and the labels of all other sentences to decide the label of the target sentence. However, the sentence embedding is generated by a convolution layer with a small filter size, which simplifies computation but limits the influence of each word.

The best two models are the LSTM-TFIDF model and the Hierarchical LSTM-W2V model (our model proposed in Section III). We believe this outcome occurred because these two models consider the following three factors: (1) the word choices, (2) the sentence positions, and (3) the relationship among sentence labels. Among these two models, the word embeddings in the Hierarchical LSTM-W2V model likely capture the semantic relationship among the words and therefore could be better input features than the TF-IDF-based features.

2) DETAILED RESULTS OF OUR MODEL

The confusion matrices of the six labels when using the Hierarchical LSTM-W2V as the prediction model are shown

in Figure 4. We also show the precision, recall, F1 score, and support (the true number of these labels) of each class in Table 6. Our model makes excellent predictions except for the “OTHERS” class. However, because the “OTHERS” class includes sentences that may have diverse and ambiguous purposes, correctly predicting the “OTHERS” class is naturally more challenging than correctly predicting the other classes. In addition, only less than 2% of the sentences belong to the “OTHERS” class. The overall performance in terms of the weighted average, micro average, and samples average is shown in the same table.

V. SYSTEM PROTOTYPE

This section introduces a prototype system to demonstrate an academic search engine understanding the purposes of each sentence and allowing users to submit queries along with their search intentions. A live demonstration site is available at <http://139.59.243.203/>.

A. SYSTEM OVERVIEW

The system is composed of an offline processing module, which is responsible for labeling the sentences in the abstracts and ingesting the extracted metadata into an indexer. The online module contains a web server to redirect user requests to the indexer, obtains the returned documents, and renders an output page to the users. An overview of the architecture is shown in Figure 5. The details of the entire system are given below.

1) OFFLINE MODULE

The offline module consists of a sentence purpose labeler, a document ingestor, and an indexer. The labeler is the Hierarchical LSTM-W2V model we introduced in Section III. As shown in Figure 5, the offline module starts by training the sentence purpose labeler based on the labeled documents (step (1) in the offline processing module in Figure 5). After training, the sentence purpose labeler labels the purposes of each sentence for each abstract (step (2)). Next, the document

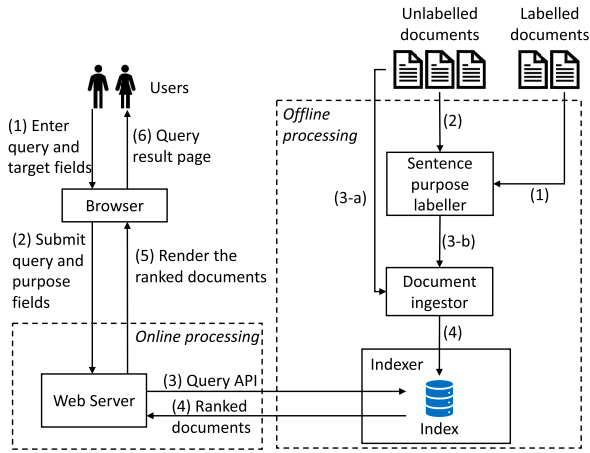


FIGURE 5. Architecture of the prototype system. The offline processing modules include the Hierarchical LSTM-W2V model to label the sentence purposes for each abstract and an indexer to digest the metadata. The online processing unit takes users’ query terms, conducts online queries to the indexed database, and returns a ranking list to the users.

ingestor takes the document metadata (e.g., titles, authors, and categories), the full text abstracts, and the label(s) of each sentence (step (3-a) and step (3-b)) and sends the information to an indexer (step 4). The indexer builds indices to facilitate the online queries. We use Elasticsearch as the indexing server.

2) ONLINE MODULE

The online module contains a web server Nginx, which receives the query term and the purpose fields from users and redirects this information to the index server (ElasticSearch) to perform a real-time query. Once the returned documents obtained from ElasticSearch, Nginx renders the web page for the users.

The user interface of the web pages are shown in Figure 6 and Figure 7. In addition to the search bar that appears in virtually all search engines, the users may select one or several fields from the six aspects to further specify their search purposes. Our model ranks the documents based on the ranking function introduced in Section V-B.

Once a user clicks on any link on the search result page, the server lists the corresponding document’s title, category, author(s), and, most uniquely, abstract and purpose(s) of each sentence in the abstract. For better representation, we use a table to indicate the relationship between a sentence and its purpose(s). As shown in Figure 7, we attach an n -by-6 table to the front of every sentence (n is the number of sentences in the abstract). Each check mark in the table indicates the existence of one aspect of a sentence. We highlight the searched terms (“support vector machine” in our example) for better visualization.

B. RANKING MODEL

This section introduces the ranking model of the prototype system. The ranking function is modified from Okapi BM25 [24].

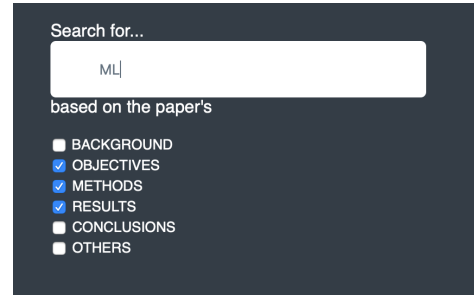


FIGURE 6. The interface for user search. We allow users to specify their target aspects.

We define the ranking score of a query term q for a document d with aspect a by Equation 15.

$$s(q, a, d_i) \propto \sum_{w \in q} (ATF(w, d_i, a) + TF(w, d_i)) \times IDF(w), \tag{15}$$

where w represents a word in query term q , $ATF(w, d_i, a)$ is called the aspect TF (ATF) score of word w in the sentences belonging to the aspects a in document d_i (Equation 16), $TF(w, d_i)$ represents the TF score of word w in document d_i regardless of the sentence aspects (Equation 17), and $IDF(w)$ is the IDF score of word w (Equation 18).

$$ATF(w, d_i, a) = \frac{f_{w,d_i,a} \times (k + 1)}{f_{w,d_i,a} + k \left(1 + b \left(\frac{|d_i|}{|D|_{avg}} - 1 \right) \right)}, \tag{16}$$

where $f_{w,d_i,a}$ returns the appearance counts of w in the sentences related to the aspects a in d_i , $|d_i|$ is the number of words in document d_i , $|D|_{avg}$ is the average number of words in a document in the corpus, and k and b are the predefined parameters with values 1.2 and 0.75, respectively, as suggested in Lucene.⁵

$$TF(w, d_i) = \frac{f_{w,d_i} \times (k + 1)}{f_{w,d_i} + k \left(1 + b \left(\frac{|d_i|}{|D|_{avg}} - 1 \right) \right)}, \tag{17}$$

where f_{w,d_i} computes the appearance counts of w in document d_i .

$$IDF(w) = \log \left(\frac{N - n_w + 0.5}{n_w + 0.5} \right), \tag{18}$$

where N is the total number of documents and n_w returns the document frequency of a word w .

The ranking model ranks the documents based on the ranking score (Equation 15) from large to small.

Compared to the classic TF-IDF measure or its variants, e.g., Lucene’s practical scoring function,⁶ the new scoring function (Equation 15) has several advantages. First, the terms $f_{w,d_i,a}$ and f_{w,d_i} in the denominators of Equation 16 and Equation 17 realize the concept of term saturation, which is when a term is more related to a document if the

⁵https://lucene.apache.org/core/8_9_0/core/org/apache/lucene/search/similarities/BM25Similarity.html

⁶https://lucene.apache.org/core/8_9_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Neural networks for stock price prediction

Categories: q-fin.ST/cs.LG/stat.ML
Authors: Song/Zhou/Han

B=background J=objective M=methods R=results C=conclusion O=others

B J M R C O

- ✓ Due to the extremely volatile nature of financial markets, it is commonly accepted that stock price prediction is a task full of challenge.
- ✓ However in order to make profits or understand the essence of equity market, numerous market participants or researchers try to forecast stock price using various statistical, econometric or even neural network models. In this work, we survey and compare the predictive power of five neural network models, namely, back propagation (BP) neural network, radial basis function (RBF) neural network, general regression neural network (GRNN), **support vector machine** regression (SVMR), least squares **support vector machine** regression (LS-SVMR).
- ✓ ✓ We apply the five models to make price prediction of three individual stocks, namely, Bank of China, Vanke A and Kweichou Moutai.
- ✓ Adopting mean square error and average absolute percentage error as criteria, we find BP neural network consistently and robustly outperforms the other four models.

FIGURE 7. The interface displaying the abstract sentences along with the corresponding label(s). This sample abstract is downloaded from <https://arxiv.org/abs/1805.11317>.

term appears more frequently in the document; however, the relevancy score saturates once the occurrence exceeds a threshold, which is effectively decided by the parameter k_s in Equation 16 and Equation 17. Second, the $|d_i|/|D|_{\text{avg}}$ term in the denominators of Equation 16 and Equation 17 accounts for the influence of the document length; that is, if a term appears the same number of times in a long document and in a short document, the short document is likely to be more relevant to the term. The b parameters in Equation 16 and Equation 17 decide the decaying speed of the relevance score as the document becomes longer. Finally, our ranking function enlarges the influence of a user's selected aspects by the ATF score. Consequently, if a term appears in some of the selected aspects of a document, the ranking score is greatly increased.

VI. DISCUSSION

This paper fulfills the two objectives listed in the introduction. First, we showed that it is feasible to apply machine learning methods to recognize the purpose(s) of each sentence in the abstract of an academic document. We compared our proposed method with eight baseline models based on 7,000 academic documents. The experimental results show that sentence position, word usage, and label relationships are crucial information for this task. Likely because our proposed Hierarchical LSTM-W2V model can effectively integrate all these types of information, the proposed model outperformed the baseline models. Second, we prototyped an academic search engine that understands the purposes of abstract sentences and enables users to query sentences with specified purpose aspects.

Initially, the task of sentence labeling may seem to fit into sequential pattern mining [8]. However, abstract texts naturally form a sequence of sequences—the word sequence

forms a sentence, the sentence sequence forms an abstract, and the sentence labels are mutually influenced. Consequently, general sequential pattern mining techniques are not the best fit for this task. It should be interesting to explore other applications that have a sequence of sequences structure and apply our proposed model to them.

Recently, attention mechanisms [30] have had tremendous success in various natural language processing tasks. An attention mechanism is flexible because it dynamically assigns weights to different parts of an input sequence. The advantage of attention is more obvious, especially when the input sequence is long, because LSTM may gradually forget earlier inputs. Therefore, a natural extension of the current model is to replace the LSTM layers with attention to form a "Hierarchical attention" model. This is one of our ongoing research directions.

Because our model can predict the purposes of each sentence in the abstract, we are currently analyzing how authors write and organize abstracts on a large scale. For example, we are interested in learning whether authors tend to introduce the background or describe the conclusion first. We also want to know whether different conferences/journals or authors from different regions prefer different writing styles in terms of how sentence purposes are organized. Our model may help answer some of these interesting questions on a large scale.

ACKNOWLEDGMENT

This work was done in part when Li-Yuan Hsu and Chia-Hao Kao were with the National Central University.

REFERENCES

- [1] C. Caragea, J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles, "CiteseerX: A scholarly big dataset," in *Proc. Eur. Conf. Inf. Retr.* Amsterdam, The Netherlands: Springer, 2014, pp. 311–322.

- [2] H.-H. Chen, L. Gou, X. Zhang, and C. L. Giles, "CollabSeer: A search engine for collaboration discovery," in *Proc. 11th Annu. Int. ACM/IEEE Joint Conf. Digit. Libraries (JCDL)*, 2011, pp. 231–240.
- [3] H.-H. Chen, M. Khabsa, and C. L. Giles, "The feasibility of investing in manual correction of metadata for a large-scale digital library," in *Proc. IEEE/ACM Joint Conf. Digit. Libraries*, Sep. 2014, pp. 225–228.
- [4] H.-H. Chen, P. Treeratpituk, P. Mitra, and C. L. Giles, "CSSeer: An expert recommendation system based on CiteSeerX," in *Proc. 13th ACM/IEEE-CS Joint Conf. Digit. Libraries (JCDL)*, Jul. 2013, pp. 381–382.
- [5] B. B. L. P. de Vries, M. van Smeden, F. R. Rosendaal, and R. H. H. Groenwold, "Title, abstract, and keyword searching resulted in poor recovery of articles in systematic reviews of epidemiologic practice," *J. Clin. Epidemiol.*, vol. 121, pp. 55–61, May 2020.
- [6] F. Deroncourt and J. Y. Lee, "PubMed 200k RCT: A dataset for sequential sentence classification in medical abstracts," 2017, *arXiv:1710.06071*. [Online]. Available: <http://arxiv.org/abs/1710.06071>
- [7] F. Deroncourt, J. Y. Lee, and P. Szolovits, "Neural networks for joint sentence classification in medical paper abstracts," 2016, *arXiv:1612.05251*. [Online]. Available: <http://arxiv.org/abs/1612.05251>
- [8] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Sci. Pattern Recognit.*, vol. 1, no. 1, pp. 54–77, 2017.
- [9] A. Fujino, H. Isozaki, and J. Suzuki, "Multi-label text categorization with model combination based on F_1 -score maximization," in *Proc. 3rd Int. Joint Conf. Natural Lang. Process.*, vol. 2, 2008, pp. 1–6.
- [10] S. Gonçalves, P. Cortez, and S. Moro, "A deep learning classifier for sentence classification in biomedical and computer science abstracts," *Neural Comput. Appl.*, vol. 32, pp. 1–15, Jul. 2019.
- [11] G. Jianping, L. Du, Z. Yuhong, and X. Taisong, "A new distance-weighted K-nearest neighbor classifier," *J. Inf. Comput. Sci.*, vol. 9, no. 6, pp. 1429–1436, 2012.
- [12] N. R. Haddaway, A. M. Collins, D. Coughlin, and S. Kirk, "The role of Google scholar in evidence reviews and its applicability to grey literature searching," *PLoS ONE*, vol. 10, no. 9, Sep. 2015, Art. no. e0138237.
- [13] A. Harzing and R. van der Wal, "Google scholar as a new source for citation analysis," *Ethics Sci. Environ. Politics*, vol. 8, pp. 61–73, Jun. 2008.
- [14] W. Huang, Z. Wu, P. Mitra, and C. L. Giles, "RefSeer: A citation recommendation system," in *Proc. IEEE/ACM Joint Conf. Digit. Libraries*, Sep. 2014, pp. 371–374.
- [15] M. J. Islam, Q. M. J. Wu, M. Ahmadi, and M. A. Sid-Ahmed, "Investigating the performance of Naive-Bayes classifiers and K-nearest neighbor classifiers," *J. Conver. Inf. Technol.*, vol. 5, no. 2, pp. 133–137, Apr. 2010.
- [16] H. Kazawa, T. Zumitani, H. Taira, and E. Maeda, "Maximal margin labeling for multi-topic text categorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 649–656.
- [17] M. Khabsa and C. L. Giles, "The number of scholarly documents on the public web," *PLoS ONE*, vol. 9, no. 5, May 2014, Art. no. e93949.
- [18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [19] M. Ley, "DBLP: Some lessons learned," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1493–1500, Aug. 2009.
- [20] Y. Liao and V. Vemuri, "Use of K-nearest neighbor classifier for intrusion detection," *Comput. Secur.*, vol. 21, no. 5, pp. 439–448, Oct. 2002.
- [21] C.-L. Liu, C.-H. Lee, and P.-M. Lin, "A fall detection system using K-nearest neighbor classifier," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 7174–7181, Oct. 2010.
- [22] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Tableseer: Automatic table metadata extraction and searching in digital libraries," in *Proc. 7th ACM/IEEE-CS Joint Conf. Digit. Libraries*, Jun. 2007, pp. 91–100.
- [23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
- [24] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to Information Retrieval*, vol. 39. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [25] S. Z. Shariff, S. A. Bejaimal, J. M. Sontrop, A. V. Iansavichus, R. B. Haynes, M. A. Weir, and A. X. Garg, "Retrieving clinical evidence: A comparison of PubMed and Google scholar for quick clinical searches," *J. Med. Internet Res.*, vol. 15, no. 8, p. e164, Aug. 2013.
- [26] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J.-Hsu, and K. Wang, "An overview of Microsoft academic service (MAS) and applications," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 243–246.
- [27] S. Tan, "An effective refinement strategy for KNN text classifier," *Expert Syst. Appl.*, vol. 30, no. 2, pp. 290–298, Feb. 2006.
- [28] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 990–998.
- [29] S. Tuarob, S. Bhatia, P. Mitra, and C. L. Giles, "AlgorithmSeer: A system for extracting and searching for algorithms in scholarly big data," *IEEE Trans. Big Data*, vol. 2, no. 1, pp. 3–17, Mar. 2016.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [31] J. Wu, K. M. Williams, H.-H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. G. Ororbia, D. Jordan, P. Mitra, and C. L. Giles, "CiteSeerX: AI in a digital library search engine," *AIMag.*, vol. 36, no. 3, pp. 35–48, Sep. 2015.
- [32] Z. Wu, J. Wu, M. Khabsa, K. Williams, H.-H. Chen, W. Huang, S. Tuarob, S. R. Choudhury, A. Ororbia, P. Mitra, and C. L. Giles, "Towards building a scholarly big data platform: Challenges, lessons and opportunities," in *Proc. IEEE/ACM Joint Conf. Digit. Libraries*, Sep. 2014, pp. 117–126.
- [33] Y. Yamamoto and T. Takagi, "A sentence classification system for multi biomedical literature summarization," in *Proc. 21st Int. Conf. Data Eng. Workshops (ICDEW)*, 2005, p. 1163.
- [34] P. Younger, "Using Google scholar to conduct a literature search," *Nursing Standard*, vol. 24, no. 45, pp. 40–46, Jul. 2010.



LI-YUAN HSU received the bachelor's degree from the Department of Computer Science, National Central University, in 2021. He is currently pursuing the degree with Texas A&M University. His research interests include machine learning and search engines.



CHIA-HAO KAO received the bachelor's degree from the Department of Computer Science and Information Engineering, National Central University. He is currently pursuing the degree with the Department of Computer Science, National Yang Ming Chiao Tung University. His research interests include computer vision and deep learning.



I-SHENG JHENG received the bachelor's degree from the Department of Computer Science and Information Engineering, National Central University. His research interests include search engine and network programming.



HUNG-HSUAN CHEN received the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, in 2013. He was a Researcher with the Industrial Technology Research Institute. He is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Central University (NCU). He is interested in data-related research topics, such as machine learning, deep learning, information retrieval, text analysis, and graph analysis. He is also interested in applying these techniques to various application domains, such as recommender systems, digital libraries, and social networks. He was a Core Member of CiteSeerX, a public search engine and digital library for academic documents.