AI Cognition in Searching for Relevant Knowledge from Scholarly Big Data, Using a Multi-layer Perceptron and Recurrent Convolutional Neural Network Model

Iqra Safder
Department of Computer Science
Information Technology University
Pakistan
iqra.safder@itu.edu.pk

Saeed-Ul Hassan
Department of Computer Science
Information Technology University
Pakistan
saeed-ul-hassan@itu.edu.pk

Naif Radi Aljohani
Faculty of Information Systems
King Abdulaziz University
Saudi Arabia
nraljohani@kau.edu.sa

ABSTRACT

Although, over the years, information retrieval systems have shown tremendous improvements in searching for relevant scientific literature, human cognition is still required to search for specific document elements in full text publications. For instance, pseudocodes pertaining to algorithms published in scientific publications cannot be correctly matched against user queries, hence the process requires human involvement. AlgorithmSeer, a state-of-the-art technique, claims to replace humans in this task, but one of the limitations of such an algorithm search engine is that the metadata is simply a textual description of each pseudocode, without any algorithm-specific information. Hence, the search is performed merely by matching the user query to the textual metadata and ranking the results using conventional textual similarity techniques. The ability to automatically identify algorithm-specific metadata such as precision, recall, or f-measure would be useful when searching for algorithms. In this article, we propose a set of algorithms to extract further information pertaining to the performance of each algorithm. Specifically, sentences in an article that convey information about the efficiency of the corresponding algorithm are identified and extracted using a recurrent convolutional neural network (RCNN). Furthermore, we propose improving the efficacy of the pseudocode detection task by using a multi-layer perceptron (MLP) classification trained with 15 features, which improves the classification performance of the state-of-the-art pseudocode detection methods used in AlgorithmSeer by 27%. Finally, we show the advantages of the AI-enabled search engine (based on RCNN and MLP models) over conventional text-retrieval models.

CCS CONCEPTS

• Information systems ~ Digital libraries and archives

ACM Reference format:

Safder, I., Hassan, S. and Aljohani, N.R. 2018. AI Cognition in Searching for Relevant Knowledge from Scholarly Big Data, Using a Multi-layer Perceptron and Recurrent Convolutional Neural Network Model. In *The 2018 Web Conference Companion (WWW 2018), April 23-27, 2018, Lyon, France*, ACM, NY, NY, 8 pages. DOI: https://doi.org/10.1145/3184558.3186334

WWW '18 Companion April 23-27, 2018, Lyon, France.

ACM ISBN 978-1-4503-5640-4/18/04.
DOI: https://doi.org/10.1145/ 3184558.3186334

1 INTRODUCTION

For decades, academics in the field of computer science have proposed algorithmic solutions to solve computational problems by creating, analyzing, and applying automated techniques such as clustering, classification, decoding, hashing, sorting, machine learning, and so on. Interestingly, a vast variety of algorithms that were originally proposed for solving a particular problem in the field of computer science have later been used to provide an efficient solution for important problems in various other fields. For instance, in bioinformatics, the alignment of nucleotide or amino acid bio sequences is modelled by using the string-matching Greedy String Tilling algorithm [1] originally proposed for plagiarism detection through text matching [2]. Similarly, Burrows-Wheeler's sequence alignment algorithm [3-4] is used extensively for DNA sequence alignment. Such an algorithm was originally proposed to undertake compression on the basis of matching repeated strings. Likewise, algorithms such as FP-growth, Prefix Span and Apriori, heavily employed in solving problems related to detecting money laundering [5-6] and in telecommunications to provide better services to customers [7], were originally proposed for finding frequent patterns in large databases Hence, with the exponential growth of scholarly big data, the task of searching for an appropriate algorithm is non-trivial.

Scholarly data consists of hundreds of thousands of research articles, many of which contain algorithms. According to Bhatia et al. [8], around 900 algorithms were published in major computer science research conferences during the years 2005 to 2009, and the number of algorithms in research articles is increasing each year. Such a phenomenon clearly shows that researchers are working actively on inventing new algorithms for emerging problems and the efficient improvement of existing deployed algorithms. There is the possibility that a new improved algorithm can improve the performance of existing systems. Therefore, software developers must keep themselves conversant with new algorithmic research horizons relating to their technologies and problems.

In the past, quite a few significant models have been proposed to search for algorithms in scholarly documents [9-10]. These models have used algorithm metadata, for instance the captions, number of lines in the pseudocode, font size and so on [11-12], in traditional search engine techniques. However, such algorithm metadata does not contain any information specific to algorithmic features, such as time complexity or the related performance metrics of precision, recall, and so on. Thus, the search is performed simply by textual matching between a user query and algorithm metadata.

Algorithmic techniques usually operate on a structured set of data, with various computational costs, to provide efficient solutions and better evaluation results for the problem in hand. An algorithm that has less computational cost yet gives improved

evaluation results is typically considered to be efficient. So far, only human experience-based heuristics are used to make a decision on the selection of a certain algorithm for a given problem, since there are no standard systematic methods in place for an appropriate algorithm search, such as on the basis of computational cost or effectiveness in terms of an algorithm's precision or recall for a given problem.

Automatic searching for algorithms and their effectiveness in scholarly big data is not a trivial task. This automatic modelling is not performed on plain document text, but on scholarly documents that contain heterogeneous document elements such as pseudocode, algorithmic procedures, tables, and images such as plots, graphs, flowcharts, and so on. These document elements are the entities, separate from the running text of the document, that are used to support and summarize the information that is written in the running text. While documenting experimental results in running text, authors add other document elements on top, either to present an argument or to summarize their discussion relating to an algorithm. Therefore, it is a challenging task to extract the evaluation results about a particular algorithm. Often, the results are summarized both in the form of supportive text and document elements. The running text that contains a discussion on the results or, more specifically, on the effectiveness of an algorithm has a context that helps the reader to understand the text. For instance, text relating to the effectiveness of an algorithm may go as follows: *"We have evaluated the LDA-SVD multi-document summarization algorithm by considering both cases of removing stop-words and not removing stop-words from the computed and the model summaries. Table 2 tabulates the ROUGE-1 recall values and its 95% confidence interval…".*

Furthermore, to model this problem comprehensively, traditional techniques such as bag-of-words, Latent Dirichlet Allocation [13], or mutual information fail to handle the semantics and word order of the text. To extract the text related to a given algorithm that conveys information about its efficiency, the text's semantics and word order are more significant, since these features are necessary to understand the context. High-order n-grams (5-gram, 6-gram, and so on) representation can be deployed, notwithstanding, to understand the context and semantics, but they suffer from a data sparsity problem that severely affects classification accuracy. In this article, we present the following three key contributions.

First, an improved machine learning-based approach was designed to enhance the accuracy of the existing baseline state-of-the-art algorithm detection approach [11] by using an MLP, a feed-forward artificial neural network model. Using the same dataset as our baseline model [11], a dataset of 258 manually annotated scholarly documents, originally selected from the CiteseerX repository, was used to validate the efficacy of the techniques deployed. Our model achieves an overall f-measure of 96.5% compared to the 75.95% reported by our baseline model [11].

Furthermore, to tap into advancements in deep learning, an algorithm [14] was deployed to create a sentence representation using both word embedding and a recurrent convolutional neural network (RCNN) for the detection of the portion of a document containing the discussion of an algorithm in terms of its effectiveness, for instance its precision, recall, and f-measure. This representation was fed into the neural network performing classification, allowing us to find accurately the '*evaluation results related text lines*' in full-text documents. Finally, we conducted experiments on the same set of 258 manually annotated scholarly documents that was earlier used for the MLP-based algorithm detection model originally obtained from CiteseerX repository. After 100 training epochs, our model achieved 76.06% accuracy.

The third contribution of this article is the prototype development of an AI-enabled search engine that implements the functionality of our proposed set of algorithms to extract further information pertaining to an algorithm's features. The accuracy of our AI-enabled search engine was compared to existing state-of-the-art models. Finally, the system was evaluated empirically to compare the effectiveness our AI-enabled search engine to conventional search engines to fetch relevant scholarly documents without the need for human cognition.

2 LITERATURE REVIEW

Although an extensive literature on scholarly information extraction is available, we discuss only important works that are closely related to ours. In addition, we discuss recent debates that augment the role of cognitive computing and tap into rigorous syntactic rules that enable a wide range of applications, not only in cognitive informatics but in web search engines, word processing, and cognitive systems in particular [15].

2.1 Brief Review of Cognitive Computing for Search Space

Our review suggests that the term "cognitive computing" is gathering popularity for systems (e.g. Watson by IBM) that intelligently process online information, beyond search, by deploying AI models to amplify existing tools to identify relevant search results from the wider scientific community. Specifically, search engines are the most common and some of the most important tools used by the general scientific community [16]. Another class of cognitive-enabled computing systems tapping into the expansion of neurologically simulated computation has recently shown potential in processing non-textual information, such as videos and online images, across disciplines [17]. Wang et al. [18] argue that computing similarity – used in many search applications, such as information retrieval, semantic web, data clustering, and natural language processing – plays a central role in many cognitive capabilities.

Further, they emphasize that cognitive-enabled models provide more highly relevant results to users' queries than the simple TF-IDF-based (term frequency–inverted document frequency) models used in many search engines. More recently, Analytis et al. [19] deployed multi-attribute utility models as cognitive search engines, including: (i) a linear multi-attribute model; (ii) equal weighting of attributes; and (iii) a single-attribute heuristic. Their experiments on 12 real-world problems, based on formal decision-making theory, show approaches that predict that how decision makers will choose, on the basis of the presented rank order, as alternatives to the one used by commercial recommendation systems and search engines.

2.2 Brief Review of Document Element Detection

The extraction of document elements such as pseudocode, algorithmic procedures, tables, figures, and plots from digital articles has been studied and explored extensively [12-13] [20-22]. Scientists often utilize such document elements in multiple ways, such as in summarizing results, describing step-by-step instructions, and illustrating ideas. Hence, the ability to detect and extract such document elements automatically would not only enable them to be indexed and searched, but also give rise to many data-mining applications that rely on these fact- and knowledge-concentrated document entities. While many of document element

extraction methods work on text-based documents, optical character recognition-based techniques have been designed for the automatic extraction of document elements [23-24]. Among these, FigureSeer [25] is prominent. This automatically extracts results from figures, using computer vision approaches, leading to the idea of the automatic extraction of results from articles. This extracted information is made available for search by efficient indexing.

A specialized table search system on ChemxSeer, has been designed for the automatic extraction of tables and figures from scholarly articles in the field of chemistry [26]. Another important search engine, AckSeer [27], has been proposed for indexing and searching acknowledgements in the CiteSeerx digital library. Note that, while both PlosOne and CiteSeerx digital libraries support a search functionality for tables and figures, none of these systems supports text summarization of document elements. To fill this gap, Bhatia and Mitra [8] propose a method to generate a summary or textual description for a document element automatically by utilizing machine learning techniques to extract and re-order the relevant sentences in the article in which the corresponding document element appears. This summarization approach helps end users to understand the relevance of a document element to their information needs.

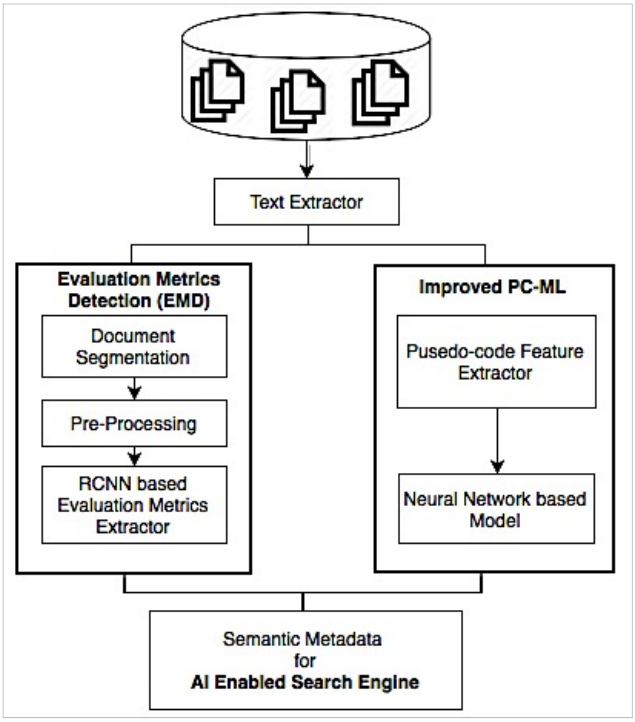

Figure 1: High-level diagram of proposed system

More recently, AlgorithmSeer [11], an algorithm search engine, was designed to undertake the automatic extraction, indexing and searching of algorithms. AlgorithmSeer has also implemented a document element summarization approach to extract algorithm textual metadata from running text. A synopsis is generated from algorithms' metadata to make them searchable on the basis of a user query. AlgorithmSeer assumes that an algorithm can be represented in pseudocode, and utilizes a machine learning-based approach to locate and extract it in a textual-represented document. While their pseudocode detection approach is sufficiently accurate to be deployed in real systems, it can be improved. Hence, we have used this approach as a baseline to propose an improved machine learning technique by using MLP neural networks.

2.3 Brief Review of Deep Neural Networks for Text Mining

Recently, the evolution in deep neural networks and representation learning has prompted new ways of solving the data sparsity problem and learning word representations [28]. Word embedding helps to measure words' semantic relatedness by using the distance between two embedding vectors. The pre-trained word embeddings and deep neural networks have proved remarkable in many Natural Language Processing (NLP) and text classification tasks. To find the semantic relatedness of phrase and sentences in scholarly documents, RCNN-based models are the most significant for text classification [20], paraphrase detection [28], semantic role labelling [29], and recursive neural tensor networks [30]. In this article, we deploy RCNN to extract the lines of text in which algorithms are discussed in terms of their effectiveness, such as their precision, recall, f-measure, and so on.

3 METHOD

In this section, we describe our proposed approach. Fig. 1 shows a high level of detail of our work to extract semantic metadata. Our approach has two sub-modules. The first presents the advancement that we have made to the existing state-of-the-art algorithm (pseudocode, or PC) detection approach [11], referred to as the *baseline PC_ML*. We trained the neural network-based model on 15 features extracted from the document. Our improved PC_ML architecture has better accuracy and precision than existing algorithms. The second module, that is, the evaluation metric detection (EMD), extracts lines of text containing a discussion of algorithms and/or experiments based on evaluation metrics. It employs a document segmentation approach [33] to extract relevant sections and then to identify the target lines by employing RCNN that has been trained on our manually tagged dataset.

3.1 Pseudocode Detection in Scholarly Articles

This section discusses our improved method for automatic detection of pseudocode in scholarly articles. Fig. 2 shows the detail of our improved pseudocode-detection approach. Our technique handles PDF documents, since a large subset of scholarly articles in digital libraries is in PDF format. First, we extracted the plain text from the PDF document by using PDFbox library (https://pdfbox.apache.org/). We also extracted the object and location information of the fonts, using the work of Hassan [31] and Tiedemann [32]. Next, a feature vector was designed, using a set of handcrafted features (see Table 1). Lastly, for classification purposes, an MLP was implemented to classify whether or not either a line of text is a pseudocode line.

Our proposed improved PC_ML method extends the baseline to improve overall accuracy and f-measure. Since machine learning-based method does not rely on the rule-based caption detection method [8], it directly detects the presence of pseudocode in articles. Normally, pseudocode is written in a spare manner that creates a sparse region in a document. Such sparse regions are called *sparse boxes*. The proposed improved PC_ML method first detects and extracts the pseudocode boxes by identifying the pseudocode lines. Next, a feature set is extracted and finally, a neural network-based classification model is applied. The following subsections contain the details of our improved PC_ML method.

3.1.1 Sparse Box Extraction. A sparse box is a set of N consecutive sparse lines. A sparse line is one that fulfills the following rules:

a) the ratio of non-space characters to average number of characters per line must be less than the threshold (0.8); b) no headers or footers; and c) encapsulated by a sparse line. This method works well with a threshold of 0.8 and N=4. Also, the method shows a high coverage of 92.99%, as per the baseline PC_ML method.

3.1.2 Feature Set Selection for Pseudocode Box Classification. A set of 15 features in four categories are extracted for each line of the sparse box. Table 1 shows the extracted features and their description: context-based (CX); content-based (CN); style-based (ST); and font style-based (FS). The CX feature captures the presence of pseudocode captions, while CN extracts the features relating to the presence of pseudocode by capturing the coding styles and pseudocode-specific keywords. ST-based features capture the sparsity of pseudocode boxes and representative symbols.

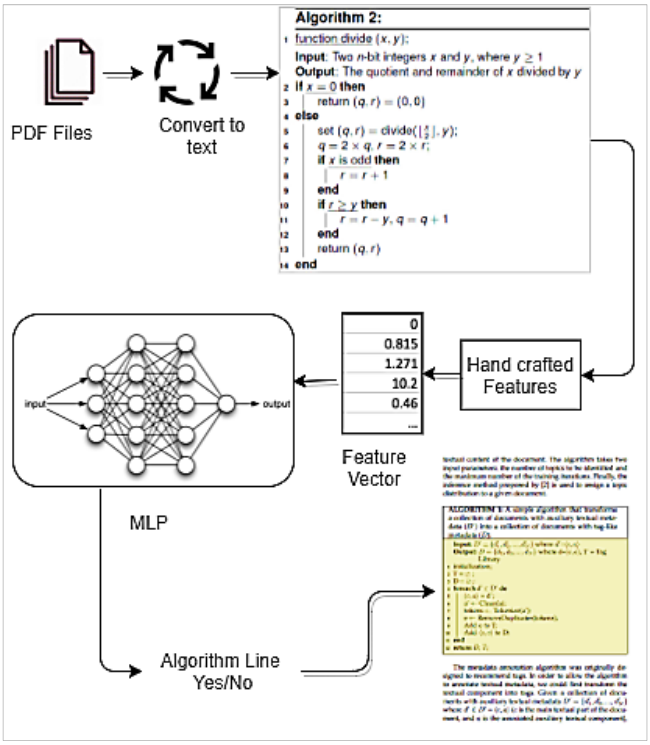

Figure 2. Improved PC_ML algorithm

3.1.3 Neural Network-based Classification Model. Each pseudocode sparse box is classified either as a pseudocode line or not by a neural network trained on the hand-tagged data. Our neural network consists of single hidden layer; the output layer consists of single neuron with sigmoid as an activation function. The input layer is of size 15 and takes the features described in Table 1 as input. The MLP classification model is trained on 70% of actual data and was tested on 30% data.

3.2 Target Text Lines Detection

This section contains details of our deployed deep learning-based algorithm that detects the target text lines relating to discussion of the evaluation of algorithms in scholarly articles. For simplicity, we refer to this method as evaluation metrics detection, or EMD. Using RCNN, the model can capture the semantics of text to classify whether or not a line is a target line. The EMD takes as input related sections of research article the sequence of words w_1, w_2, w_3...w_n and outputs the class of the text. The probability function p(k|D, θ) function is used to find the probability of text line belonging to a class containing target lines. The following pre-processing steps were taken:

3.2.1 Hierarchical Section Extractor. Generally, articles are organized into sections. To extract evaluation result lines from research articles, we need to segment an article into its standard sections (i.e. abstract, introduction, background, related work, and so on). The purpose of segmentation is to keep only those sections that have high chance of containing results-related discussion. We used a rule-based method to segment out the standard sections of an article; this technique helped us to keep only the related sections (i.e. methodology, results, experiments, abstract, etc.) and discard unrelated sections in which the chance of target lines is minimal or close to none (i.e. introduction, related work, references, acknowledgements, etc.). Afterwards, text cleaning was performed to remove header/footers, the article title, author affiliations, and so on. Lastly, the text of the cleaned and related sections was given as input to the RCNN model.

Table 1: Feature set for pseudocode (PC) classification, categorized into: content-based (CN); structure-based (ST); font style-based (FS); and context-based (CX)

	Feature	Description
CN	PC keywords %	Ratio of PC words and no. of words
	PC symbols %	Ratio of PC symbols and no. of chars in line
	Word sparsity	Ratio of no. of words and avg. no. of words in a line
	Char. sparsity	Fraction of no. of chars in a line and avg. no. of chars
ST	Greek chars %	Ratio of Greek symbols and no. of chars in line
	Comment sign %	Ratio of comment signs and no. of lines
	Functions %	Ratio of no. of functions and no. of lines
	Begins with a no.	Whether or not line begins with a no.
FS	Mode font size	Mode font size of text
	Variance font size	Variance font size of text
	Font styles %	No. of font styles (combinations of font style and font name)
	Font-style switches	No. of font-style switches
	Indentation	Whether or not the line is indented
	Avg. indentation of first 4 characters	Avg. indentation of first 4 chars
CX	Is it a caption	Whether or not the text line is a caption line

3.2.2 Deep Learning-based Neural Network Model. Fig. 3 shows the RCNN [14] structure. The model takes the words and the context as a representation of a word. We used a bidirectional RCNN to capture the context of words. For the left and right context of word wi, Vectors cr(wi) and cl(wi) are defined, cl(wi), computed using Eq. 1. Here, W(l) is a matrix that is used to transform context between the hidden layers. W(sl) is also a matrix, and is used to combine the left word's context with the current word. While e(wi-1) is the word-embedding vector of word wi-1, with real value elements, cl(wi-1) is the left-side context of the previous word wi-1. Similarly, cr(wi) is calculated in same manner, as shown in Eq. 2.

$$c_l(w_i) = f((W^{(l)})c_l(w_{i-1}) + (W^{(sl)})e(w_{i-1})) \quad (1)$$

$$c_r(w_i) = f\left((W^{(r)})c_r(w_{i-1}) + (W^{(sr)})e(w_{i-1})\right) \quad (2)$$

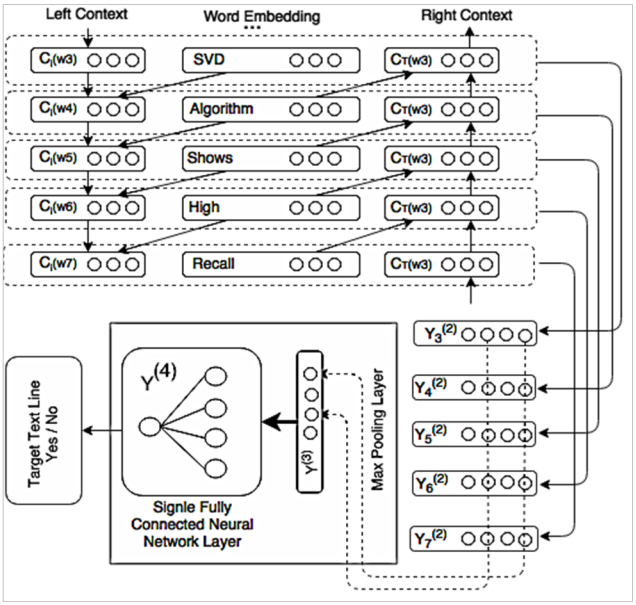

Figure 3. RCNN network structure with subscripts to denote the position of a word in a sentence

Eq. 1 and Eq. 2 are context vectors for left and right words. Referring to Fig. 3, $c_l(w_7)$ contains the context and semantics of all left-side words in the form of encodings. The left context of the network is "SVD algorithm shows high" with all previous words of sentence. Similarly, $c_r(w_7)$ contains the right context. Next, the word w_i representation is learned by combining left and right contexts, as shown in Eq. 3:

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \quad (3)$$

The model obtains the all c_l and c_r in forward and backward passes, respectively. After learning the x_i representation of a word, a linear transformation was applied, with a *tanh* activation function to add some nonlinearity:

$$y_i^{(2)} = tanh(W^{(2)}x_i + b^{(2)}) \quad (4)$$

The $y_i^{(2)}$ is a latent semantic vector that contains the most important and powerful factors for the text representation by analysing each and every semantic factor. The model also contains a max-pooling layer, as mentioned in the network architecture (see Fig. 3). The pooling layer converts the different length text to a fixed length vector. It helps to find the most important information through the entire text. Therefore, after learning the all words' representation, we applied a max-pooling layer:

$$y^{(3)} = max_{i=2}^{n} y_{(i)}^{(2)} \quad (5)$$

The max function is applied element wise, and the maximum of the *k-th* element of $y_{(i)}^{(2)}$ is in the *k-th* position of $y^{(3)}$. Finally, the model contains a single fully connected hidden layer as an output layer, as shown in Eq. 6:

$$y^{(4)} = W^{(4)}y^{(3)} + b^{(4)} \quad (6)$$

Finally, a sigmoid activation function was applied to $y^{(4)}$ to give us the probability number, as shown in Eq. 7:

$$p_{(i)} = \frac{exp(y_k^{(4)})}{1+exp^{(y_k^{(4)})}} \quad (7)$$

Training of RCNN Model: First, we defined all the model parameters for training θ, as shown in Eq. 8:

$$\theta = \{E, b^{(2)}, b^{(4)}, c_l(w_1), c_r(w_n), W^{(2)}, W^{(4)}, W^{(l)}, W^{(r)}, W^{(sl)}, W^{(sr)}\} \quad (8)$$

Here, the vectors $b^{(2)}, b^{(4)}$ are real-valued vectors and E is real-valued word embeddings, whereas $W^{(2)}, W^{(4)}, W^{(l)}$, $W^{(r)}$, and $W^{(sl)}, W^{(sr)}$ are transformation matrices and $c_l(w_1), c_r(w_n)$ are initial left and right context real-valued vectors. The target was to maximize the probability (log likelihood) with respect to θ.

$$\theta \rightarrow \sum_{c \in D} log\ p(class_c | D, \theta) \quad (9)$$

In Eq. 9, D is the set of documents and $class_c$ is the positive class of text data. We used the stochastic gradient descent (SGD) for training optimization.

Further, we employed Skip-gram model-based word embeddings (commonly used in NLP tasks [33]) for word representation. Since our dataset suffers from a class imbalance problem, to avoid bias in our model we included the following balancing techniques: a) *Random Over-sampling (ROver)*, in which minority class examples are randomly replicated until both classes become equal; and b) *Random Under-sampling (RUnder)*, in which majority class examples are randomly dropped until both classes become equal. Note that the same 70% data was used for training and 30% for testing, as for the PC_ML model. However, after applying the balancing techniques, we had 4337 positive samples; that is, the target lines and 4770 negative instances, namely text lines other than target text lines, containing no information about the efficiency of the corresponding algorithm. Finally, we adjusted the network hyper parameter settings, such as H (hidden layer size) to 100, learning rate to 0.001, V (vocabulary size) to 3000, and training epochs to 100.

3.3 AI-enabled Search Engine Using Improved PC_ML and EMD Models

Our AI-enabled search engine has the following steps: i) synopsis generation using improved PC_ML and EMD models for each document in the system; ii) standard indexing for both the dataset, that is, the synopsis, and full text documents; and iii) finally, deployment of a state-of-the-art searching model to rank the results against user queries for a comparative analysis of a simple full text-based corpus and a synopsis-based corpus.

4 EXPERIMENTATION AND EVALUATION

The experiments comprised two modules: pseudocode detection using MLP; and target text line detection (conveying information about the efficiency of the corresponding algorithm) using RCNN. The experiments were performed on Ubuntu, Nvidia Titan 750 GPU with 2 GB memory. We used the Python Chainer Library (https://chainer.org/) for RCNN implementation and Weka (https://cs.waikato.ac.nz/) for MLP implementation.

The dataset consists of 258 scholarly articles selected from the CiteSeer^x repository [11]. Note that our baseline model PC_ML used the same dataset, making our improved PC_ML comparable with the baseline. This dataset consists of 275 pseudocodes and 282 unique algorithms. For target text line detection, we used same dataset of 258 scholarly articles. Note that, of the total, that is, 37,000 text lines in our dataset, only 6.3% contain target text that conveys information about the efficiency of the corresponding algorithm. Note that the annotation was undertaken by four human experts, who identified 2331 text lines as target lines.

4.1 Evaluation of Improved PC_ML Detection

Standard precision, recall, f-measure and accuracy indices are used for the evaluation of pseudocode detection and target text line detection.

Table 2 shows the comparison of our baseline PC_ML and improved PC_ML for pseudocode detection. We found that the improved PC_ML (using MLP) outperformed PC_ML (baseline) across all evaluation metrics. We achieved a 98.8% f-measure with the baseline improved PC_ML model, compared to only 75.95% with our PC_ML model. This is a 27% improvement on the baseline. Similarly, we achieved a significant improvement in terms of recall (from 67.17% to 96.7%), precision from (87.37% to 97.4%), and overall f1 measure (from 75.95% to 96.5%), respectively.

4.2 Evaluation of the EMD Model

For the evaluation of EMD model on our dataset, we used the following experimental settings. The network hyper parameters were assigned as follows: hidden layer size (H) 1000; learning rate 0.01; vocabulary size (V) 3000; and training epochs 100. The training calculating the metrics for every class and taking the average accuracy of our EMD method for 100 epochs was to depict the behaviour of our model during training. In the first 20 epochs, the model started learning very quickly. Afterwards, it showed a gradual increase in accuracy and reach, up to 76.06% (see Fig. 5). Next, we presented our testing results in terms of precision, recall, and f-measure. The RCNN-based EMD model achieves 0.77 precision, 0.73 recall, and 0.75 f-measure, which is a very reasonable way to detect target lines in this type of database. Note that we reported the weighted precision, recall, and f-measure scores by calculating the metrics for every class and taking the average.

Table 2: Precision, recall, and f-measure for pseudocode-detection model

Method	Model	Re%	Pr%	F1%
PC_ML	Baseline	67.17	87.37	75.95
PC_ML (improved)	MLP	96.7	97.4	96.5

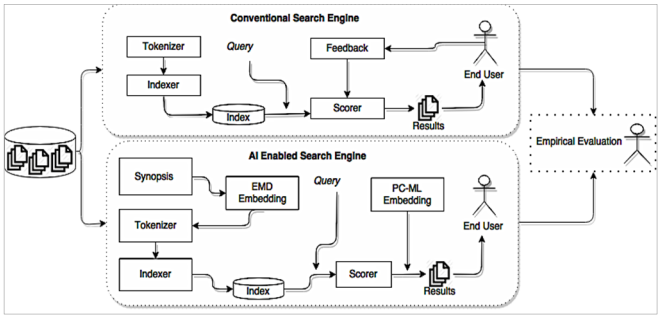

Figure 4: System architecture of proposed AI-enabled search engine

5 PROTOTYPE OF AI-ENABLED SEARCH ENGINE

In this section, a prototype of an AI-enabled search engine is presented. Fig. 4 illustrates the high-level architecture of our proposed system. The corpus of 30% testing data, that is, 74 scholarly articles – containing 82 pseudocodes and 95 unique algorithms – is used to evaluate empirically the AI-enabled search engine system results against a conventional search engine.

First, we enriched synopses generated from documents by embedding target text lines, as detected by the EMD model. Next, these enriched synopses were tokenized and indexed to make them searchable. User queries were employed to obtain a ranked list of results. The top-matched algorithms, along with their synopses, were presented to the end user. Lastly, algorithms detected by PC_ML were embedded in the query search results to make these more comprehensive and elaborative. Using Okapi BM25 [34], similarity scores were computed to rank the query search results for both search engines. In order to evaluate empirically the performance of our proposed AI-enabled search engine against a state-of-the-art conventional search engine, we selected a set of queries and relevant documents that were identified by two independent human annotators.

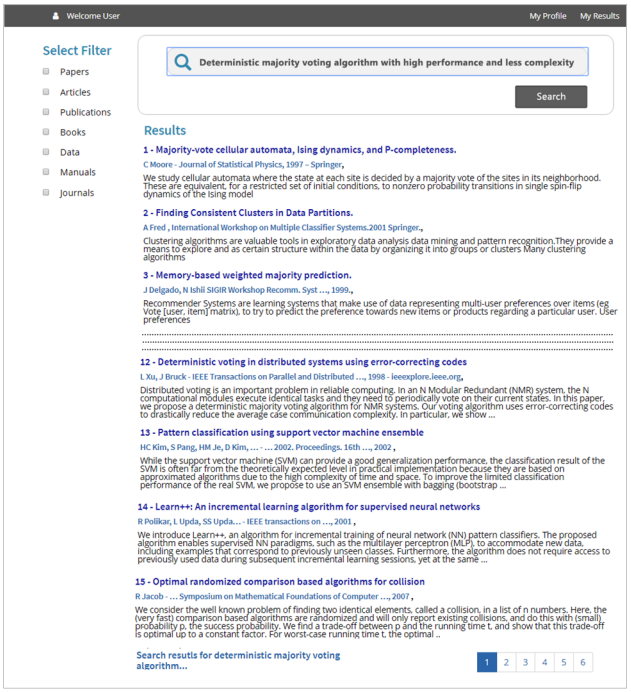

Figure 5. Screenshot of search results from conventional search engine

Fig. 5 shows the top 15 documents for the search query *"Deterministic Majority Voting Algorithm with High Performance and Less Complexity"*, using a conventional search engine. Note that the deterministic majority voting algorithm is used for N-modular redundancy. A detailed analysis revealed that the top 10 documents on the list present a discussion of this algorithm along with a comparative analysis of different algorithmic techniques. We also found a few documents that discuss the voting techniques used in classification-related problems and have nothing to do with the deterministic majority voting algorithm. Interestingly, the document that originally presented the algorithm and provided a detailed discussion on its performance evaluation appears in 12th position.

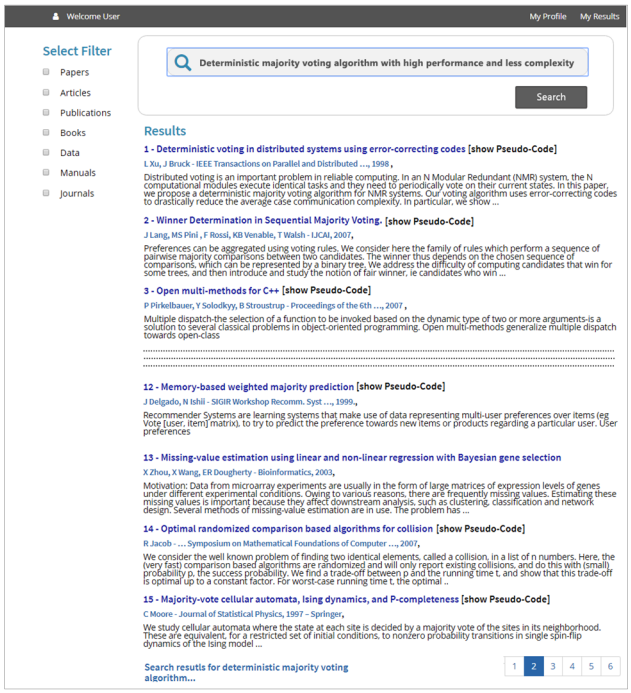

Figure 6: Screenshot of search results from AI-enabled search engine

In contrast, the search results from our AI-enabled search engine appear to be very different. Fig. 6 shows the top 15 documents for the same query. A detailed analysis of these documents showed that the top five documents present either pseudocode or an algorithmic procedure. Interestingly, our desired document, which appeared 12th in the conventional system, now appeared first. Further, PC_ML embedding allows the user to click on the respective "Show Pseudo-Code" link to actually see the section of the article where the pseudocode(s) have been placed in the full-text document. Overall, this offers a great user experience and enables users to search for the relevant information quickly. Since the indexing and searching techniques are implemented to undertake a text-based document retrieval task, searching for algorithms is a non-trivial problem that demands specialized knowledge to narrow down to the specific required algorithm. Therefore, the main purpose of this section is to demonstrate a basic prototype search system that utilizes algorithmic specific metadata, including performance-related features such as precision, recall, f-measure, and so on, in order to improve the relevance of the search results.

6 CONCLUDING REMARKS

Computer science is all about developing and applying appropriate algorithms to computational problems. Recently, a search engine for algorithms that are represented as pseudocode has been proposed and prototyped. However, such an algorithm search engine matches only the user query to the extracted generic textual metadata of each pseudocode. In this article, we proposed solutions to enhance the performance of the pseudocode-detection task by training a multi-layer perceptron classifier using 15 features that specifically characterize the composition of pseudocode in scholarly documents. The proposed algorithm improves on the state-of-the-art pseudocode-detection method by 27%. Furthermore, we deployed state-of-the-art word embedding and an RCNN model to discover and retrieve sentences from a document to convey the information about the efficiency (such as precision, recall, and f-measure) of the corresponding algorithm. Finally, using a prototype of our AI-enabled search engine, we showed that our deployed models can improve search results. As future work, we plan to overcome some of the limitations of this work, such as the deployed technique using the same embedding vector for numeric figures and English-language text.

Therefore, for the following text pertaining to algorithmic feature: *"the given method outperforms base model by 20% with precision 81.5 and recall 81.5...."*, we could employ natural language processing and machine learning techniques to extract a numeric representation of algorithmic performance. This information will enable a direct comparison to other algorithms. Furthermore, we could investigate the possibility of extracting other algorithm-specific metadata, such as run-time complexity, input, output, and compatible data structures, in a large-scale data corpus.

REFERENCES

[1] M. J. Wise, "Neweyes: a system for comparing biological sequences using the running Karp-Rabin Greedy String-Tiling algorithm.," in *ISMB*, 1995, pp. 393–401.

[2] Z. DJurić and D. Gašević, "A source code similarity system for plagiarism detection," *Comput. J.*, vol. 56, no. 1, pp. 70–86, 2013.

[3] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *ArXiv Prepr. ArXiv13033997*, 2013.

[4] P. J. Ochieng, T. Djatna, and W. A. Kusuma, "Tandem repeats analysis in DNA sequences based on improved Burrows-Wheeler transform," in *Advanced Computer Science and Information Systems (ICACSIS), 2015 International Conference on*, 2015, pp. 117–122.

[5] Z. Chen, A. Nazir, E. N. Teoh, E. K. Karupiah, and others, "Exploration of the effectiveness of expectation maximization algorithm for suspicious transaction detection in anti-money laundering," in *Open Systems (ICOS), 2014 IEEE Conference on*, 2014, pp. 145–149.

[6] R. Dreżewski, G. Dziuban, Lukasz Hernik, and M. Pączek, "Comparison of data mining techniques for Money Laundering Detection System," in *Science in Information Technology (ICSITech), 2015 International Conference on*, 2015, pp. 5–10.

[7] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens, "New insights into churn prediction in the telecommunication sector: A profit driven data mining approach," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 211–229, 2012.

[8] S. Bhatia and P. Mitra, "Summarizing figures, tables, and algorithms in scientific publications to augment search results," *ACM Trans. Inf. Syst. TOIS*, vol. 30, no. 1, p. 3, 2012.

[9] S. Bajracharya *et al.*, "Sourcerer: a search engine for open source code supporting structure-based search," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 681–682.

[10] C. McMillan, M. Grechanik, D. Poshyvanyk, C. Fu, and Q. Xie, "Exemplar: A source code search engine for finding highly relevant applications," *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1069–1087, 2012.

[11] 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, 2016.

[12] S. Tuarob, S. Bhatia, P. Mitra, and C. L. Giles, "Automatic detection of pseudocodes in scholarly documents using machine learning," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, 2013, pp. 738–742.

[13] S. Hingmire, S. Chougule, G. K. Palshikar, and S. Chakraborti, "Document classification by topic labeling," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 877–880.

[14] Safder, I; Sarfraz, J; Hassan, SU; Ali, M; Tuarob, S; Detecting Target Text related to Algorithmic Efficiency in Scholarly Big Data using Recurrent Convolutional Neural Network Model, 19th International Conference on Asia-Pacific Digital Libraries (ICADL), Bangkok, Thailand.

[15] Wang, Yingxu, and Robert C. Berwick. "Formal relational rules of english syntax for cognitive linguistics, machine learning, and cognitive computing. "*Journal of Advanced Mathematics and Applications* 2, no. 2 (2013): 182-195.

[16] Gil, Yolanda, et al. "Amplify scientific discovery with artificial intelligence." *Science* 346.6206 (2014): 171-172.

[17] M. Martialay, "Citizen Scientist," The Approach; http://approach.rpi.edu/2014/04/25/citizen-scientist-your-safari-photos-are-the-data/.

[18] Wang, Y., Rolls, E.T., Howard, N., Raskin, V., Kinsner, W., Murtagh, F., Bhavsar, V.C., Patel, S., Patel, D. and Shell, D.F., 2015. Cognitive Informatics and Computational Intelligence: From Information Revolution to Intelligence Revolution. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *7*(2), pp.50-69.

[19] Analytis, P.P., Kothiyal, A. and Katsikopoulos, K.V., 2014. Multi-attribute utility models as cognitive search engines. *Judgment and Decision Making*, *9*(5), pp.403-419.

[20] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification.," in AAAI, 2015, vol. 333, pp. 2267–2273.

[21] T. Mikolov, W. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations.," in *Hlt-naacl*, 2013, vol. 13, pp. 746–751.

[22] B. Coüasnon and A. Lemaitre, "Recognition of Tables and Forms," in *Handbook of Document Image Processing and Recognition*, Springer, 2014, pp. 647–677.

[23] S. Z. Chen, M. J. Cafarella, and E. Adar, "Searching for statistical diagrams," *Front. Eng. Natl. Acad. Eng.*, pp. 69–78, 2011.

[24] S. Kataria, W. Browuer, P. Mitra, and C. L. Giles, "Automatic Extraction of Data Points and Text Blocks from 2-Dimensional Plots in Digital Documents.," in *AAAI*, 2008, vol. 8, pp. 1169–1174.

[25] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "FigureSeer: Parsing Result-Figures in Research Papers," in *European Conference on Computer Vision*, 2016, pp. 664–680.

[26] P. Mitra, C. L. Giles, B. Sun, Y. Liu, and A. R. Jaiswal, "Scientific Data and Document Processing in ChemxSeer.," in *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, 2008, pp. 51–56.

[27] M. Khabsa, P. Treeratpituk, and C. L. Giles, "Ackseer: a repository and search engine for automatically extracted acknowledgments from digital libraries," in *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, 2012, pp. 185–194.

[28] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning, "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection.," in *NIPS*, 2011, vol. 24, pp. 801–809.

[29] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[30] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, 2013, vol. 1631, p. 1642.

[31] T. Hassan, "Object-level document analysis of PDF files," in *Proceedings of the 9th ACM symposium on Document engineering*, 2009, pp. 47–55.

[32] J. Tiedemann, "Improved text extraction from PDF documents for large-scale natural language processing," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2014, pp. 102–112.

[32] S. Tuarob, P. Mitra, and C. L. Giles, "A hybrid approach to discover semantic hierarchical sections in scholarly documents," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, 2015, pp. 1081–1085.

[33] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.," in *ACL (1)*, 2014, pp. 238–247.

[34] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M. and Gatford, M., 1995. Okapi at TREC-3. *Nist Special Publication Sp*.

[35] Rose, S., Engel, D., Cramer, N. and Cowley, W., 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pp.1-20.