Information Processing and Management 57 (2020) 102269

Contents lists available at ScienceDirect

Information Processing and Management

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

Deep Learning-based Extraction of Algorithmic Metadata in Full-Text Scholarly Documents

Iqra Safder[a], Saeed-Ul Hassan[a], Anna Visvizi[b], Thanapon Noraset[c], Raheel Nawaz[d], Suppawong Tuarob[c,*]

[a] Information Technology University, 346-B, Ferozepur Road, Lahore, Pakistan
[b] Deree College - The American College of Greece, 6 Gravias Street, 153-42 Aghia Paraskevi, Athens, Greece
[c] Faculty of Information and Communication Technology, Mahidol University, Thailand
[d] Department of Operations, Technology Events and Hospitality Management, Manchester Metropolitan University, Manchester, United Kingdom

ARTICLE INFO

Keywords:
Knowledge-based Systems
Algorithmic Metadata
Algorithm Search
Deep Learning
Bi-Directional LSTM
Information Retrieval
Full-text Articles

ABSTRACT

The advancements of search engines for traditional text documents have enabled the effective retrieval of massive textual information in a resource-efficient manner. However, such conventional search methodologies often suffer from poor retrieval accuracy especially when documents exhibit unique properties that behoove specialized and deeper semantic extraction. Recently, AlgorithmSeer, a search engine for algorithms has been proposed, that extracts pseudo-codes and shallow textual metadata from scientific publications and treats them as traditional documents so that the conventional search engine methodology could be applied. However, such a system fails to facilitate user search queries that seek to identify algorithm-specific information, such as the datasets on which algorithms operate, the performance of algorithms, and runtime complexity, etc. In this paper, a set of enhancements to the previously proposed algorithm search engine are presented. Specifically, we propose a set of methods to automatically identify and extract algorithmic pseudo-codes and the sentences that convey related algorithmic metadata using a set of machine-learning techniques. In an experiment with over 93,000 text lines, we introduce 60 novel features, comprising content-based, font style based and structure-based feature groups, to extract algorithmic pseudo-codes. Our proposed pseudo-code extraction method achieves 93.32% F1-score, outperforming the state-of-the-art techniques by 28%. Additionally, we propose a method to extract algorithmic-related sentences using deep neural networks and achieve an accuracy of 78.5%, outperforming a Rule-based model and a support vector machine model by 28% and 16%, respectively.

1. Introduction

Rapid growth in scholarly documents has been observed due to increasingly enormous research activities conducted by scientific communities both in academia and industry over time (Xie et al., 2019; Shardlow et al., 2018; Hassan et al., 2018; Xia et al., 2017). The advancements in web-based tools and technologies have enabled us to store and index millions of articles in digital archives.

* Corresponding Author.
E-mail addresses: iqra.safder@itu.edu.pk (I. Safder), saeed-ul-hassan@itu.edu.pk (S.-U. Hassan), avisvizi@acg.edu (A. Visvizi), thanapon.nor@mahidol.edu (T. Noraset), r.nawaz@mmu.ac.uk (R. Nawaz), suppawong.tua@mahidol.edu (S. Tuarob).

https://doi.org/10.1016/j.ipm.2020.102269
Received 5 November 2019; Received in revised form 2 April 2020; Accepted 14 April 2020
Available online 30 May 2020
0306-4573/

Academic search engines and digital libraries such as Google Scholar[1], Web of Science[2], Microsoft Academic[3], and Semantic Scholar[4] have become a necessary means for researchers to keep abreast of scientific advancements (Safder & Hassan 2019; Ananiadou, Thompson & Nawaz 2013). The search mechanisms used for these scholarly repositories typically represent a document as a bag-of-words without sophisticated semantic processing and rely on simple keyword-based search and textual matching techniques, which do not effectively interpret query/document semantic relevance in certain information needs (Vo & Bagheri 2019; Sinoara et al., 2019; Hassan et al., 2017). For instance, to answer the query "Which model gives the best accuracy on IMDB data set?", simple text-matching the query and document text may be insufficient. This is because most text-based information retrieval systems tend to return documents that contain the query terms, which may not even discuss any algorithms that were used on the IMDB dataset. Hence, this kind of sophisticated information requires intelligent methods that match the content of a scientific paper with relevant algorithm-specific metadata. However, most traditional information retrieval approaches fail to deliver precise answers to such specific algorithmic needs, since they treat a document as a 'chunk of full-text', which is later processed as a 'bag-of-words'. Such document processing fails to extract crucial information that would satisfy complex and specific queries (Azad & Deepal 2019; Wang et al., 2011; Arshad et al., 2019)

In computer science and other computing-related fields, researchers publish their newly developed algorithms and corresponding metadata information in scholarly documents, disseminated through conference meetings, journal publications, open-access archival services, etc. A significant number of these scientific articles contain pseudo-code (algorithm) along with relevant algorithmic contexts such as evaluation results of an algorithm, datasets used by that algorithm and its runtime complexity. Bhatia and Mitra (2012) estimated that roughly 850 algorithms were published in top-tier computer science conferences during 2005-2009. The ability to identify these newly emerging algorithms and to extract their corresponding metadata could prove crucial for developing a specific, semantic-enhanced search methodology for algorithms extracted from a vast and ever-increasing pool of scholarly documents (Li & de Rijke 2019; Imran et al., 2018)

Searching for a well-known algorithm along with its algorithmic specific metadata, (time complexity, dataset, and results) is quite easy, as these "standard" algorithms are typically already indexed and made available online. Generally, we consider an algorithm as a standard algorithm if it is eminent and recognized by its name, such as the Heapsort algorithm, Dijkstra's algorithm, Bellman-Ford algorithm, and Floyd–Warshall algorithm, etc. Such standard algorithms and their corresponding metadata are manually collected and cataloged in algorithm textbooks, Wikipedia, and specialized websites for computer programmers such as Tutorialpoint[5] and Programming-Algorithm.net[6]. According to an initial survey in 2010, Wikipedia.com cataloged around 1,700 standard algorithms. Furthermore, The National Institute of Standards and Technology (NIST) has a personalized collection of over 289 standard algorithms. While standard algorithms are well managed and cataloged, newly proposed algorithms in scientific publications are not as well curated, simply because they are both too new, and hence not yet well received by research and practice communities. Furthermore, the massive emergence of these newly invented algorithms published in scientific documents inherently introduces challenges to manual curation of these algorithmic inventions and their metadata.

Searching for novel algorithms is non-trivial as it requires a tedious effort of manual searching scientific papers and reading through them until the user finds the desired algorithmic information. However, having to read through entire research articles would not only make it a tiresome procedure of finding the desired algorithms, but also could discourage many novice algorithm searchers from seeking cutting-edge algorithmic solutions and diverge away to more standard, yet not so ground-breaking algorithms which are much easier to find. Moreover, this process may become more exasperated if the searchers are non-experienced in scientific article search who have the tendency to use inappropriate searching keywords. Hence, automatic identification and extraction of algorithms along with their metadata could significantly support a successful algorithm search mechanism. Recently, a prototype algorithm search engine has shown improvements to mine algorithms and algorithmic specific metadata from full-text articles (Tuarob et al., 2016). While these techniques can extract algorithm representations (as in pseudo-codes) and generate a textual summary for each algorithm, many aspects are still left for improvement and investigation. Specifically, their pseudo-code extraction method could be significantly improved by incorporating features that characterize pseudo-code content. Furthermore, algorithm-specific metadata could be extracted from the document proposing an algorithm. For instance, pseudo-codes corresponding to algorithms and their associated properties, such as the dataset statistics, performance, and runtime analysis would be additional useful to algorithm seekers when searching for desired algorithms (Bakar, Safder & Hassan 2018).

In this research, we first observed that pseudo-codes are written in different font sizes and multiple styles, separated from the body of text in a scholarly article. Therefore, tailored feature engineering techniques that capture pseudo-code characteristics are investigated for automatic algorithm identification. Moreover, we observed that only a small portion of content in an algorithm proposing paper discusses the properties of such an algorithm. Table 1 shows notable examples of sentences that convey algorithmic properties, found in the discussions related to the evaluation results of an algorithm, dataset for experimentations and run-time complexity of an algorithm. In this research, we aim to identify and extract such sentences to provide richer algorithmic metadata to extracted algorithms. Since the algorithmic metadata is usually composed of textual context with varying writing styles followed by

[1] https://scholar.google.com
[2] www.webofknowledge.com
[3] https://academic.microsoft.com
[4] https://www.semanticscholar.org
[5] https://www.tutorialspoint.com/
[6] https://www.programming-algorithms.net/

Table 1
Example of sentences conveying algorithmic metadata. [E = Evaluation, D = Dataset, C = Complexity, G = General]

Algorithmic specific metadata sentences	Class Label
"The experiments indicate that the ARG editing distance, although the slowest, has certain advantages over all other methods: (a) It is the most accurate demonstrating higher precision and recall than any other method, followed by Hungarian methods and 2D strings." (Petrakis & Georgiadis 2000).	E
"The data for this task consists of 500 news articles from the New York Times and Associated Press Wire services each paired with 4 different human-generated reference summaries (not actually headlines), capped at 75 bytes." (Rush et al., 2017).	D
"For a sorted list of weights, this algorithm runs in $O(Hlog\ log(n/H))$ time with $0(n)$ work, where H is the length of the longest generated code." (Milidiú et al., 1999).	C
"Assigning a category to a given word (tagging) depends on the particular word and on the categories (tags) of neighboring words. A theory that is able to assign tags to a given text is then a recursive logic program by nature." (Jorge & de Andrade Lopes 1999).	G

different authors and fields of study, using keyword-based approaches to identify algorithmic related sentences may not be sufficient to capture semantics related to algorithmic metadata.

In this paper, we propose a set of novel machine learning techniques to identify algorithms represented by pseudo-codes and to extract algorithm-specific metadata from a scientific document. Moreover, we present a prototype of an algorithm-focused search system with an emphasis on finding algorithms proposed in the scientific documents. Such a system leverages the proposed machine learning approaches to extract algorithms and algorithmic metadata, which could be used to enhance semantic annotation to corresponding algorithms, hence further improving the semantic search capability. The main contributions of this paper are as follows:

We propose a novel machine learning-based method to discover algorithms (represented as pseudo-codes) in scientific articles. We frame the problem into a binary classification task, where a text line is classified whether it is part of a pseudo-code or not. A total of 60 novel features are extracted from each text line in the document, divided into three subcategories: Font Style based (FS), Content-Based (CN), and Structure-based (ST). Furthermore, we deploy five machine learning classifiers such as Random Forest (RF), KNN, Naïve Bayes, Logistic Regression and Decision Tree to classify each text line. The proposed approach is validated on a dataset of 256 research papers, randomly downloaded from CiteSeerX repository. We found that RF achieved 93% F1-score, outperforming the existing state-of-the-art techniques by 28%.

We propose a deep neural network-based classification technique to identify algorithmic metadata sentences that discuss algorithmic performance evaluation, datasets used, and runtime complexity. We frame this problem into a classification task where a sentence in an algorithm proposing paper is classified into algorithmic related classes. The proposed approach uses character level embedding to produce word-level streams that effectively classify these contextually enriched algorithmic metadata sentences by treating each type of algorithmic metadata as a separate class. Furthermore, we perform a set of rigorous experiments to validate our proposed technique against two baselines: the Rule-based and a Support Vector Machines (SVM) classifier. The input dataset contains roughly 37,000 sentences extracted from 256 scholarly articles, originally downloaded from CiteSeerX repository. The proposed deep learning approach outperforms the Rule-based and SVM methods by 56.69% and 25.80% respectively, in terms of accuracy.

We present a prototype, algorithm-focused search system, Enhanced-IR System, that incorporates our algorithm metadata extraction method. We deployed a case study using 21,940 full-text publications originally downloaded from the Association of Computational Linguistics (ACL) repository. First, we generate a specialized algorithmic-based summary called document synopsis for each full-text document. Furthermore, we enriched these synopses by inserting algorithmic metadata extracted by our proposed deep learning model. Lastly, we use Apache Lucene to index the extracted algorithmic metadata and make them searchable. We empirically evaluate the efficacy of the proposed Enhanced-IR system against a conventional TF-IDF based system, which only indexes the whole textual content in each document.

The remaining paper is organized as follows. In Section 2, we discuss related work. In Section 3, we discuss our dataset and proposed methodology using machine learning and deep learning models. Section 4 explains the experimentation and our achieved results in detail followed by a comprehensive discussion about our proposed deep learning-based search engine. Lastly, in Section 5, we present a conclusion and discussion on future directions and potential applications of this research.

2. Literature Review

Extensive research has been done to investigate the effective retrieval of different document elements such as tables, figures, and pseudo-codes from the full body text of an article (Rastan, Paik, Shepherd 2019; Safder & Hassan 2018). Different Rule-based and machine learning-based techniques were proposed and deployed for these documents' element retrieval tasks (Altınel, B., & Ganiz 2018; Safder, Hassan & Aljohani 2018; Safder et al., 2017). Prominent among these is the AlgorithmSeer (Tuarob et al., 2016), i.e. an algorithm search engine, which automatically extracts algorithms from research papers by means of a hybrid machine learning approach. The system utilizes an ensemble machine learning method, which learns from 47 features automatically-extracted from each sparse region in a document. The classification model is then used, first, to classify each region whether it is a pseudo-code or not, and then, second, the system extracts useful textual information corresponding to these algorithms by utilizing a synopsis generation method. They also presented a case study on the heterogeneous pool of 20,000 research papers, to improve the search results when search queries are algorithm-related.

Moreover, a specialized chemistry search engine, *ChemxSeer* (Mitra et al., 2007), was proposed for the extraction of figures, tables and corresponding metadata. The system assists chemists and researchers to search for chemical formulae written in indexed chemistry publications. Later, they also designed *TableSeer*, a search system for tables, along with this work. *AckSeer* (Khabsa, Treeratpituk & Giles 2012) is an acknowledgment search engine built on the *CiteSeerX* digital library. It uses regular expressions to extract acknowledgments from full-text documents. Entities are then extracted from acknowledgments using named entity recognition (NER) tools. Recently, methods for extracting and understanding mathematical expressions have been proposed. An example of such applications is document classification based on mathematical expressions written in academic documents, which categorizes mathematical documents into their mathematical category, e.g. graph theory, probability theory, and linear algebra (Suzuki & Fujii 2017).

To satisfy certain IR queries, information extracted from images, plots, and graphs presented in the articles could prove essential. A few attempts have been made to extract information out of these plots by separating the axis, their labels, legends and data points using a computer vision approach ((Al-Zaidy and Giles, 2017; Choudhury, Wang & Giles 2016). Such efforts have established that extracting figures from PDF files is a challenging task, mainly due to insufficient labeled datasets to train accurate models. This is made possible by inducing labels to a large dataset and training the neural network for figure extraction (Seigel 2018; Clark & Divvala 2016). A state-of-the-art approach for figure extraction, FigureSeer, was proposed, which implements a ResNet based approach to mine information from figures in scholarly documents (Seigel 2016). The algorithm first localizes potential figure candidates and then classifies each of them whether it is a figure or not. This approach also handles clutters and deformation, resulting in a better, more accurate and more powerful search. Meuschek (2018) designed an adaptive approach for plagiarism detection from independent text components such as images, diagrams, and flowcharts from academic documents. The proposed method incorporates state-of-the-art image analysis techniques with image similarity assessment and position-aware OCR based techniques. The achieved results indicate that this advanced technique can complement the content-based feature analysis approach for academic plagiarism detection.

More recently, few attempts were made to develop semantic relations between entities. Al- Zaidy and Giles (2018) proposed an unsupervised model to construct the entity-relationship graph by avoiding manual data tagging for training. This construction is done by an iterative approach where a semantic learner learns from previously extracted tuples. The nodes represent entities, and thus a relationship can be developed for better analysis. These graphs were used for algorithmic discovery, or to rank the documents based on their relationship in the graph (Xiong 2017).

Text classification is a prominent area of research for many years. Traditional and classical machine learning models such as SVM, Naïve Bayes, k-Nearest Neighbors are among the standard choices for document classification tasks (Joachims 1998; Rubin et al., 2012, Mahmood et al., 2020). A prominent drawback of these models is that efficiency and performance rapidly decrease for multi-label classification. Recent advancements in representation learning and deep neural networks have devised new ways of learning textual representations (Nguyen, Duong & Cambria 2019; Le & Mikolov 2014). Learning representation of words and sentences from the data, deep learning models, such as CNN (Kim 2014), RNN (Lai et al., 2015), RCNN (Sundermeyer, Schlüter & Ney 2012) and LSTM (Greff, 2016), appears to be highly effective in text classification, in contrast with traditional bag-of-words based models.

Zhang et al. (2015) introduced a multi-class text classification using character-level convolutional neural networks on large-scale datasets. The methods were evaluated against the traditional models, such as BOW, N-grams, TF-IDF, and deep leaning word-based CNN and LSTM models. They found that the baseline models overcame the proposed deep learning models with small sample sets; whereas, in the case of large-scale datasets, character level deep learning approaches were superior. Inspired by computer vision deep networks, Conneau et al. (2016) proposed deep architectures for text classification. The networks operate directly at the character level and employ small convolutions and pooling layers. They also debated that the deep convolutional layers increase the performance of deep learning models for text classification.

Recently, Mai et al. (2018) proposed a document title-based semantic subject indexing approach using deep learning MLP, CNN and RNN models. They claimed that their proposed deep learning models outperformed the available full-text systems by a large margin. Safder et al. (2018) designed a metadata sentence classification approach using an LSTM deep learning model. Moreover, other studies have focused on encoding syntactic knowledge (part of speech) to enhance sentence and phrase-level representation for text classification (Huang, Qian & Zhu, 2017). Regardless, usages of deep learning techniques for the extraction of metadata specific to document elements (such as algorithms) in scholarly documents have not been fully explored.

3. Methodology

We propose a set of machine learning based techniques to automatically identify pseudo-codes (PCs) and extract algorithmic metadata sentences from a scientific document. We frame both the problems into text classification problems where each text region/sentence is classified whether it is a pseudo-code/an algorithmic metadata sentence or not. The section presents the detail of our proposed approaches, as illustrated at a high level in Fig. 1. Firstly, we converted the PDF files into text files. Secondly, we extracted algorithmic PCs present in the scientific papers. Afterward, we extracted algorithmic metadata present in those papers that at least contain one or more algorithmic PCs. Lastly, we present a case study of an advanced search system that utilizes the proposed algorithm-specific metadata extraction to improve algorithm-focused search results.

3.1. Algorithmic pseudo-code extraction

In this section, we discuss the details of our designed machine learning-based technique to extract PC from research articles (see

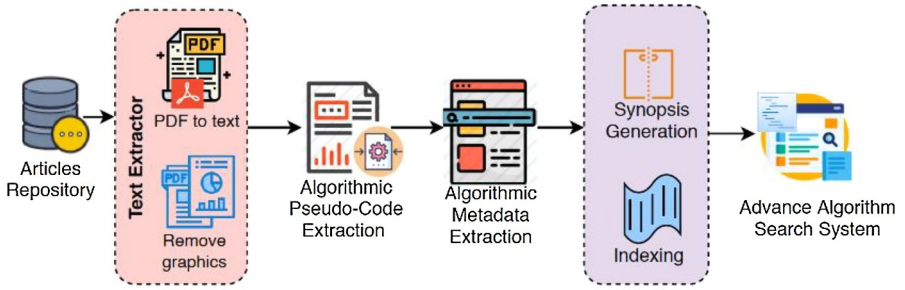

Fig. 1. High-level system architecture.

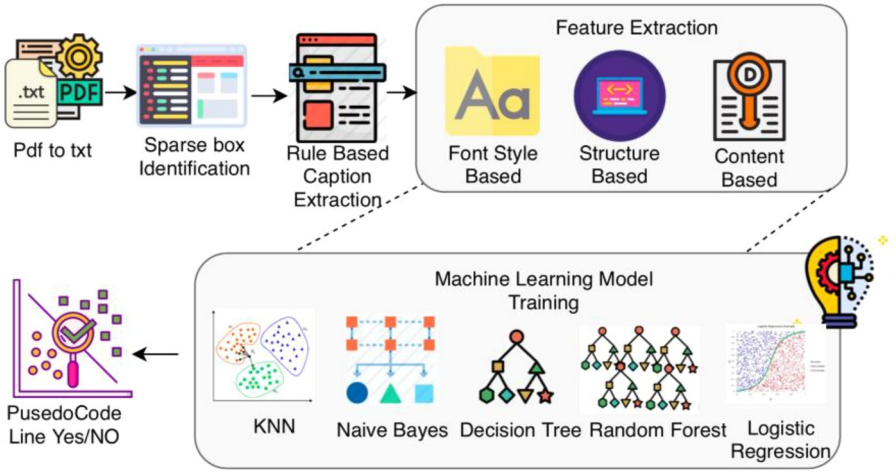

Fig. 2. Proposed mechanism for algorithmic PC extraction.

Fig. 2). Firstly, we converted PDF articles into text documents using Java-based Apache PDFBox[7] API. It supports extraction of unicode text and eliminates graphical images from a PDF document. Inspired by the work of Hassan (Hassan 2009), we made some modifications to PDFBox's library to also extract object information such as font styles, font sizes and text locations from a PDF document. After this process, only textual content and their font-information are retained, while figures and text alignment information are removed to facilitate subsequent text mining algorithms. Then, 60 features are extracted from each text line, which are used to train classification models to identify PC lines in a given scientific paper.

Generally, scientific publications articulate their proposed algorithms in the form of PCs. A PC is defined as a set of step-by-step instructions for solving a computational problem. Note that, mathematical expressions are not considered PCs, as they do not represent step-wise computational instructions. These PCs are considered as document elements that are separated from the running text, having distinguishable identifiers such as captions, function names, and algorithm names. Note that, since PCs can appear anywhere in a scientific publication, their identifiers (captions, functions names and algorithm names) serve as their references in the running text. Here, we present a set of novel features that extract font style based, structure-based and context-based features from PCs to segregate them from the body of the running text in scientific publications. The following is the detail of the designed model.

3.1.1. Sparse region identification

To extract PCs from scholarly documents, we first captured the sparse regions in documents with the observation that not all PCs are accompanied by PC identifiers such as captions. Therefore, a regular expression based technique that merely focuses on detecting PC captions tend to suffer from low recall because these non-captioned PCs would remain undetected (Tuarob et al., 2016).

Furthermore, we analyzed PCs present in our dataset and found that 25.8% of the total PCs do not have any accompanying PC captions. We started with the initial observation that a PC is typically written in a sparse manner (named as a sparse region) in a document. Fig. 3 shows samples of sparse regions on a PDF document and corresponding raw-text pages.

A sparse box is defined as a set of M successive sparse lines. Whereas a sparse line is a line that fulfills the following conditions: The line should not be a header or footer line.

The ratio of the non-spaced characters C with an average number of characters per line $AvgC$ must be lesser than the threshold z, that is $\frac{C}{AvgC} < z$.

[7] https://pdfbox.apache.org/

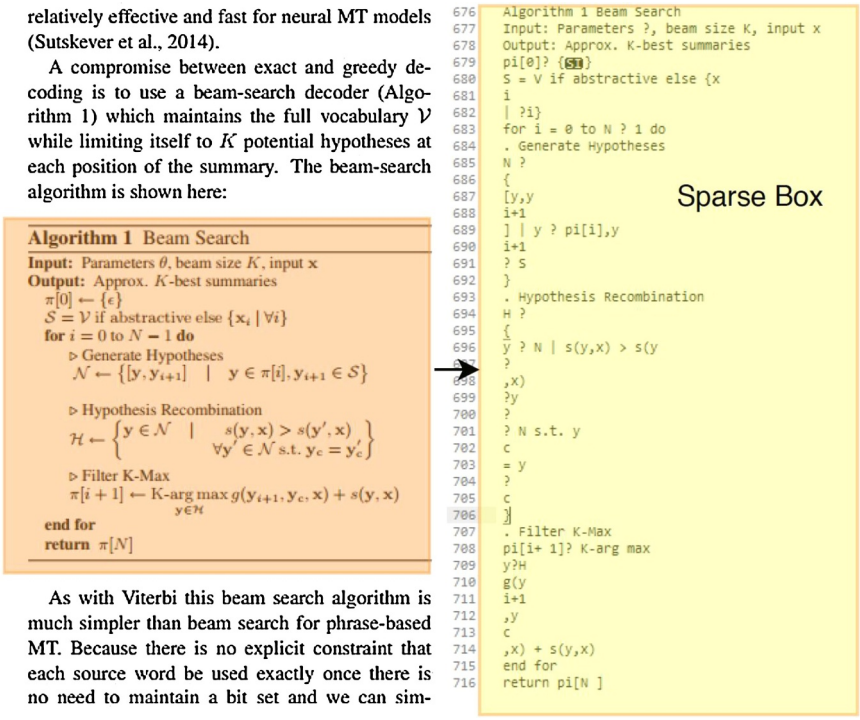

Fig. 3. An example of sparse region extraction for algorithmic PC extraction

The four text lines upper or lower the given line must be a sparse line.

3.1.2. Feature engineering for PC identification

We compiled a heterogeneous group of features that contain structure-based (ST), content-based (CN) and font style based (FS) features. Table 2 presents the list of the extracted features along with their descriptions. We extracted 60 features comprising 18 CN, 34 ST, and 8 FS features. The CN feature set captures the captions and the presence of algorithm-related keywords. Similarly, the FS based features examine the variations in font styles, font sizes, and indentation used to compose a PC. The ST features analyze the text sparsity along with the character types and symbols used. Furthermore, we deployed multiple feature normalization techniques to avoid model over fitting.

3.1.3. Algorithmic Metadata Extraction

The proposed algorithmic metadata extraction model leverages the capacity of deep learning classification techniques to identify text lines in a given document that discusses metadata information of the proposed algorithmic PC within. Specifically, we employed a character convolutional neural network and two Bi-Directional LSTM architectures to classify each sentence in an article into one of the three algorithmic metadata classes, or non-algorithm related class. Note that, word embedding based techniques (Mikolov et al.,2013) are well adapted for domain-specific tasks, but acquiring prior knowledge is nevertheless a costly process – it requires a predefined word vocabulary, and the structural parser needs to deal with many different variations including morphological shifts and undefined chunking. Such specifications make text comprehension more or less specialized to a particular language – in case of language modifications, many rules need to be built from scratch.

Additionally, deep learning systems are capable enough to understand text scripts without artificially integrating external knowledge about words, sentences or any other language-related syntactic and semantic concepts (Zhang & LeCun 2015). Interestingly, in the English language, all dictionary words are generated by 26 alphabets (or 52 in case of both upper and lower case alphabets or a few more special characters). Therefore, with character embedding, any word vector can be generated even from out-of-vocabulary words. Whereas, word embeddings can only handle seen words. Likewise, character embeddings work well for misspelled and new words. Moreover, it manages rare words much better than word2vec because word embedding vectors suffer from the lack of training data for infrequent words. Additionally, in the case of character-based models, there are a small number of vectors; therefore, they are quite efficient in terms of model time complexity (Kim et al., 2016). Fig. 4 explains the detailed architecture of our proposed approach. Firstly, the PDF documents are parsed to extract textual content. Next, we performed the document segmentation and pre-processing to clean noisy text remnants and classify each chunk of text into corresponding sections. Furthermore, the cleaned sentences are fed to the deep learning model for classification of algorithmic specific metadata. Lastly, we combined the extracted algorithm metadata sentences with sentences that directly reference an algorithm (e.g., "*Our proposed approach is summarized in Algorithm 2.*") for efficient document searching.

Table 2
Feature list for pseudo-code extraction.

	Feature	Description
CN	Word Sparsity	Ratio of no. of words to avg. no. of words in a line
	Character Sparsity	Ratio of number of chars in a line to avg. no. of chars
	No. Algo Keywords	No. of algorithm keywords (e.g algorithms, pseudo-code, etc.) in a line
	Percent Algo keywords	No. of algorithm keywords in a line to the number of words.
	Percent Algo. Keywords to lines	Ratio of algorithm keywords to the number of lines.
	Number of PC keywords	Number of PC keywords (e.g. if, else, for, for all, while, do, iftrue, ifend etc.) present in a line
	Percent PC keywords	No. of PC keywords in a line to the number of words.
	Percent PC Keywords to lines	Ratio of PC keywords to the number of lines.
	Begin with PC Keyword	Is line begin with a PC keyword or not
	No. lines begin with PC Keyword	Number of lines begin with a PC keyword.
	Percent of begin PC keywords	Ratio of the number of lines begins with a PC keyword to the number of lines.
	Is Function	Is a function detected by regex, e.g. FindMin(), present in the line or not
	Number of Functions	Total number of functions in a line e.g. count(s)
	Functions Percent	Ratio of no. of functions to no. of lines
	Is PC caption	The current line is a PC caption or not.
	PC Label Nearby	Is the line has a PC label (e.g. Algorithm Z, Procedure, etc.) nearby in a window (three lines above or below the line)
	PC Caption Nearby	Is a PC Caption present within a window range.
	Caption Nearby	Is a general caption or PC caption present within the window range.
ST	Begin with a Digit.	Is line begin with a digit
	Lines begin with a Digit	Total number of lines begins with a digit.
	Percent Lines begin with a Digit	Ration of the number of lines with a digit to the total number of lines.
	Lines End with a Dot	Total number of lines end with a dot.
	Percent Lines End with a Dot	Ratio of the number of lines end with a dot to the number of lines.
	Chars Count	Number of chars in a line.
	Percent Chars	Ratio of the number of characters to the number of lines.
	Symbol Count	Number of symbols present in a line.
	Percent Symbol Count	Ratio of number symbols present in a line to the number of characters.
	Alphabets Count	Number of alphabets in a line.
	Percent Alphabets Count	Ratio of the number of alphabets in a line to the number of characters.
	Digit Count	Number of digits present in a line
	Percent Digit Count	Ratio of the number of digits to the number of characters.
	Alphanumeric Count	Number of alphanumeric characters present in a line
	Percent Alphanumeric	Ratio of number of alphanumeric characters to the number of characters
	Non Alphanumeric Count	Number of non-alphanumeric characters present in a line
	Percent Non Alphanumeric	Ratio of the number of non- alphanumeric characters to the number of characters
	Greek Chars Count	Number of Greek characters present in a line.
	Percent Greek Chars	Ratio of the number of Greek characters to the number of characters
	Arrows Char Count	Number of arrows chars in a line
	Percent Arrows Char	Ratio of number of arrow chars to the number of characters
	MathOps Count	Number of math operators in a line
	Percent MathOps Count	Ratio of the number of math operators to the number of characters.
	Comment Signs Count	Number of comment signs in a line
	Percent Comment Signs	Ratio of number of comment signs to the number of characters
	PC Symbol Count	Number of PC symbols (e.g. math operators, arrows, comment signs)in a line
	Percent PC Symbol	Ratio of Number of PC symbols (e.g. math operators, arrows, comment signs)in a line to the number of chars
	1CharWords Count	Number of 1character words in a line
	Percent 1CharWords	Ratio of number of 1character words in a line to the number of lines
	Percent 1CharLines	Ratio of number of lines with 1character words to the total number of lines
	ijk Chars Count	Number of ijk characters in a line
	Percent ijk Chars	Ratio of number of ijk characters in a line to the number of lines
	Coding Symbol Count	Number of coding symbols in a line
	Percent Coding Symbol Count	Ratio of the number of coding symbols in a line to the number of lines
FS	Mode FS	The mode font size of a line
	Variance FS	The Variance value of the font size of a line.
	Indentation	Is the text line is indented or not
	Indentation Variance	Indentation Variance value of a text line.
	Avg. Indentation Of First 4 Chars	Average indentation of first 4 characters in a line.
	Number of Diff FS	The total number of font sizes present in a line.
	Number of FS Switches	Count the number of font style switches in a line
	FS Switches Percent	Ratio of number of font style (different combination of font style and font name) to the number of lines.

3.2.1. Deep Neural Network Model

The proposed model scans through each sentence in a document, then classifies it into an algorithm metadata type. Firstly, the text character streams are fed to the sentence encoding component to generate sentence level embedding. Note that, the LSTM is added at the encoding stage to keep track of word and character positions in a sentence embedding. Furthermore, these learned sentence encodings are passed to another Bi-Directional LSTM to capture sentence context and semantics. Lastly, we added a layer of neurons as an output layer. The details of the components in the model architecture are discussed in the following sections.

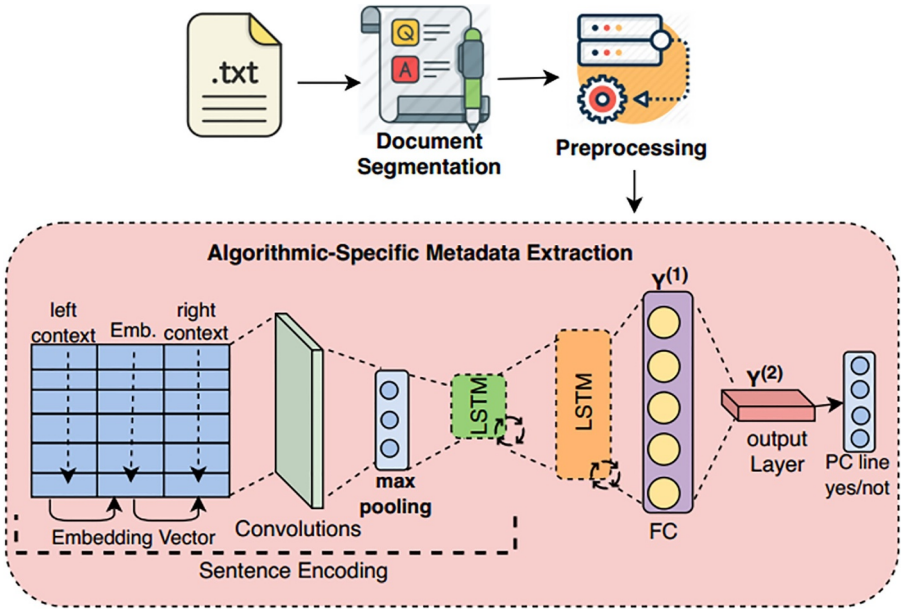

Fig. 4. Detailed architecture of the algorithmic metadata extraction model.

3.2.2. Hierarchical Section Extraction

Generally, a scholarly article is organized into eight standard sections, namely, Abstract, Introduction, Background, Proposed Methodology, Experiments, Results, Conclusions, and References. We observed that the location of the sentences can provide useful signals for identifying algorithmic metadata. For example, discussions related to the results of an algorithm are usually added into the Experimentation & Results section. Similarly, discussions related to algorithmic time complexity and datasets are typically added into specific sections of an article. Therefore, we employed a document segmentation approach (Tuarob, Mitra & Giles, 2015) to separate the document into standard sections. Note that the relevant sections such as Abstract, Methodology, Experimentation & Results and Conclusion, which are likely to contain discussions related to algorithmic metadata are segregated from all irrelevant sections (Introduction, Background, and References). Moreover, we also applied text cleaning techniques using heuristic rules and regular expression to get rid of junk lines (lines that are part of text remnants from tables or lines that do not form complete words), header & footers, authors name and affiliations, and document titles. In the end, the processed text is provided to the designed classification model.

3.2.3. Sentence Encoding and Classification

Our designed deep neural network accepts character one-hot encoding schemes as an input vector. The vector is generated by selecting m characters as language alphabets, where $m = 70$ (including 26 English alphabets, 10 digits, new line and 33 other special characters) is the number of the possible character set. Then, each character is represented by a one-hot vector (size of m). Note that characters outside of the set, such as space, are encoded as zero vectors. The output is 16 dimensions vector per every single character.

At the embedding stage, firstly, the characters (one-hot encoding) are read by the CNN (convolution and max pool layer) to generate a sequence of 0/1 vectors representing the sentence. Since sentences may have different lengths, to avoid inconsistent representation from CNN, we employ the LSTM layer to summarize the sequence of vectors into a single fixed-size vector. Specifically, the sequence of vectors from the CNN is fed to two bidirectional LSTM layers, and the concatenation of the last output states (both directions) is the encoded sentence. The final representation is a vector intended to capture the semantics of a sentence without losing ordering information of words (characters) and to model dependencies of words (characters) in both forward and backward directions. To classify a sentence, we apply fully-connected layers to transform the LSTM output into a probability distribution over the classes. The class with the highest probability is the output of our classification model.

3.2.4. Detailed Model Architecture

Our model has the following detail. For the character-level CNN, we use one convolutional layer with 256 filters with a filter width of 3 (three characters are read at a time), followed by a max-pooling layer with a kernel width of 21. For the bidirectional LSTM layers, we use two layers having 256 and 128 hidden units respectively. Lastly, the fully connected layers also contain two layers. The first fully-connected layer has 100 units and uses a *ReLU* activation function as presented in Equation 1:

$$y_i^{(1)} = ReLU\left(W^{(1)}x_i + b^{(1)}\right) \tag{1}$$

where x_i is the encoded representation of input data learned by the LSTM. The final layer has a *softmax* activation function to compute the probability distribution of the output classes as shown in Equation 2 and Equation 3.

$$y^{(2)} = W^{(2)}y^{(1)} + b^{(2)} \tag{2}$$

$$p_{(i)} = \frac{\exp(y_k^{(2)})}{\sum_{k=1}^{n} \exp^{(y_k^{(2)})}} \tag{3}$$

For regularization, we applied a dropout with probability = 0.3 in between the layers.

4. Results and Discussion

In this section, we discuss the details of our experimentation and evaluation results for PC extraction and algorithmic metadata extraction. We also present a simple baseline method called a Rule-based technique and a machine learning method to compare their results with our proposed algorithmic metadata extraction model. Lastly, we present a prototype of an enhanced scholarly search engine to illustrate the efficacy and applicability of our proposed approaches.

4.1. Datasets

We used two datasets in this research, one for pseudo-code identification and algorithmic metadata extraction, and the other for evaluating our enhanced search engine.

The first dataset consists of 256 PDF articles containing 275 pseudo-codes, and 282 unique algorithms, originally downloaded from CiteSeerX repository. These PDF documents were randomly selected from the set of documents that contain at least one occurrence of the term 'algorithm'. For pseudo-code (PC) identification, we extracted 93,565 text lines from the given dataset and performed manual annotation to tag them as PC or non-PC lines. We found that only 8% of the total lines are labeled as PC lines (Class 1), whereas the other 92% are non-PC lines (Class 0). Out of 256 papers, only 92 papers contain one or more PCs along with their associated textual metadata. Therefore, these 92 papers were used for the algorithm metadata extraction validation. We preprocessed the 92 research articles to remove textual remnants and junk lines. Finally, we are left with 37,000 sentences, among which 7% of sentences are annotated as the positive class (algorithmic metadata sentence) and the remaining 93% sentences are tagged as negative class (non-algorithmic metadata sentence) instances. Note that, 7% (2,577) of the positive instances comprise three types of algorithmic specific metadata 1) 2,331 sentences discussing algorithmic efficiency, 2) 136 sentences relating to datasets in a given document, and 3) 80 sentences describing algorithmic time complexity. We used this dataset to evaluate the performance of our proposed deep learning technique for algorithmic metadata extraction.

To implement a search system, a larger set of documents is required. The second dataset consists of 21,940 articles, downloaded from ACL, containing 1,500 unique algorithms. We used this dataset as a case study to evaluate the effectiveness of our proposed enhanced searching system. The complete list of Paper_IDs used in the second dataset can be downloaded from https://github.com/slab-itu/algo-metadata.

4.2. Evaluation of Algorithmic Pseudo-Code Extraction

We implemented a set of machine learning classifiers including Random Forest (RF), Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Decision Stump (DS), Linear Logistic regression (LLR), Support Vector Machines (SVM), Naïve Bayes (NB), k-Nearest Neighbors (KNN), Logistic Regression (LG), and Decision Tree (DT). However, we only reported the results for the top five best performing classifiers such as RF, NB, KNN, LR, and DT because the prelim results for the remaining classifiers were not good compared to other tree-based classifiers due to the rule-extracted features. For model training, we used five-fold cross-validation for all classifiers. For RF, we set *criterian = gini* to balance class weights. Moreover, we computed Mean PR-curves and Feature-based PR-curves to measure the effectiveness of each feature type. These curves elaborate on the model performance and behavior with all features and for each feature group.

Fig. 5 shows the PR-curves for five best performing classifiers along with their AUCPR values using five-fold cross-validation. We also computed feature-based PR curves for each feature group to get insights about the impact of each feature type on the overall performance. We found that RF outperformed all the classifiers with 0.98 AUCPR using all features. Moreover, DT secured 0.91 AUCPR and KNN achieved 0.78 AUCPR. LG and NB remained the lower performing classifiers among all with 0.65 and 0.09 AUCPR respectively. Note that NB is a conditional probability model that works well with discrete attribute values. Additionally, it also suffers from zero conditional probability issue (division by zero). Therefore, it underperformed when dealing with continuous variables in our dataset.

Additionally, Table 3 shows the precision, recall, and F1-score for all the deployed classifiers using 5-fold cross-validation. The results indicate that RF is the best performing model because of its ability to handle multiple decision trees in parallel. The result is the average of all the decision trees. Since a single decision tree has a high value of variance that tends to over-fit easily, the averaging mechanism (or voting) helps to reduce the variance and avoid model over-fitting. Moreover, RF selects subsets of data to learn, therefore; it works well even on large datasets. Additionally, it has the ability to handle class imbalance by attempting to minimize the overall error rate to optimize model performance.

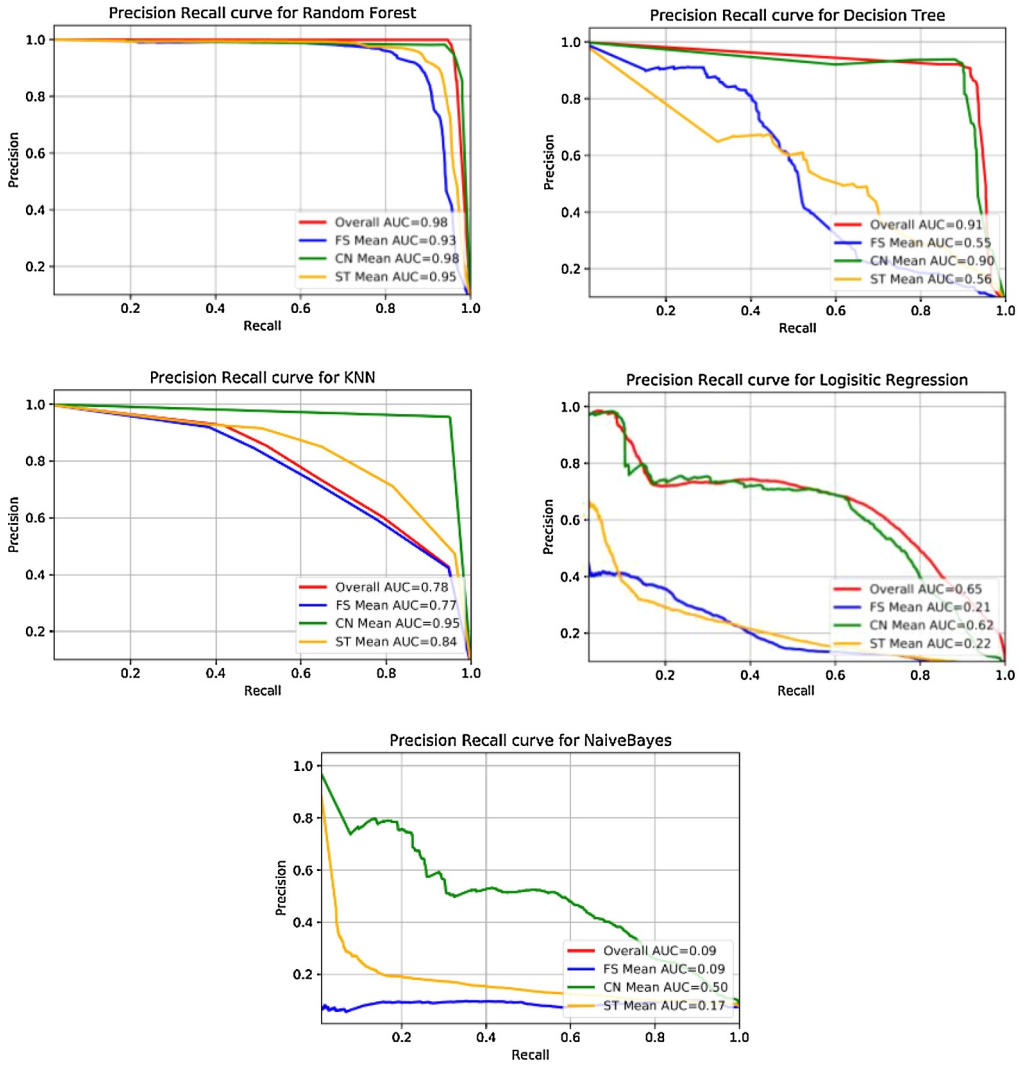

Fig. 5. Mean and feature based PR curves for all classifiers.

Table 3
Precision, recall, and F1-score for all classifiers.

Classifier	Precision	Recall	F1-score
Random Forest	**0.96**	**0.91**	**0.93**
Decision Tree	0.73	0.87	0.80
KNN	0.66	0.62	0.64
Logistic Regression	0.25	0.89	0.37
Naïve Bayes	0.09	0.68	0.15

We also present the feature-based PR-curves for all feature groups; CN, FS, and ST, for each classifier (see Fig. 5). We observed that CN based features are the best performing feature set among all feature groups with 0.98 AUCPR. This is because such features are capable of capturing PC contents, including algorithm/PC keywords and the presence of captions. Moreover, FS and ST features have also shown promising results independently.

Furthermore, we have also compared our best performing classifier RF trained with our proposed features using five-fold cross-validation with existing state-of-the-art pseudo-code extraction methods (See Table 4). Overall our PC extraction model (PC-ML60) has outperformed the existing state of the art (Taurob et al., 2016) Rule-based (PC-BL), machine learning (PC-ML) and combined (PC-ML) PC extraction techniques with 90.02% precision, 90.76% recall and 93.32% F1-score, outperforming the best baseline (PC-CB) by +9.9% precision, +35.12% recall and +22.87% F1-score respectively.

Table 4
Precision, recall, and F1-score for PC extraction.

Method	Model	Precision%	Recall%	F1-Score%
PC-BL (Tuarob et al., 2016)	Baseline	70.46	35.96	47.62
PC-ML (Tuarob et al., 2016)	Majority-Voting (LMT-RF-RIPPER-MLR)	85.31	57.04	68.04
PC-CB (Tuarob et al., 2016)	Majority-Voting (LMT-RF-RIPPER)	87.37	67.17	75.95
PC-ML60	Majority-Voting (RF)	**96.02**	**90.76**	**93.32**

4.3. Evaluation of Deep Neural Network for Algorithmic Metadata Extraction

To train a deep neural network model, we optimize the network using categorical cross-entropy loss. We set the number of epochs to 100. Further, to overcome the class imbalance problem in our input data, we deployed standard data balancing techniques such as random undersampling and oversampling. Moreover, we performed a set of experiments using multiple optimizers such as Adam, Adamax, Adagrad, Adadelta and RMSprop with the learning rate = 0.001.

Fig. 6 depicts the learning accuracy and loss behavior during the model training using different optimizers. The model has shown a very smooth behavior for optimizers Adam and RMSProp. Whereas, the other three optimizers (Adagrad, Adadelta, and Adamax) did not perform well for our dataset. Their accuracy/loss curves have shown irregular behavior during the model training. Overall, Adam and RMSprop optimizers yielded the best performance. In the first 80 epochs, we observed a rapid increase in training accuracy and later, we have seen a gradual increase in training accuracy until it reached up to 95% with a loss value approaching to zero.

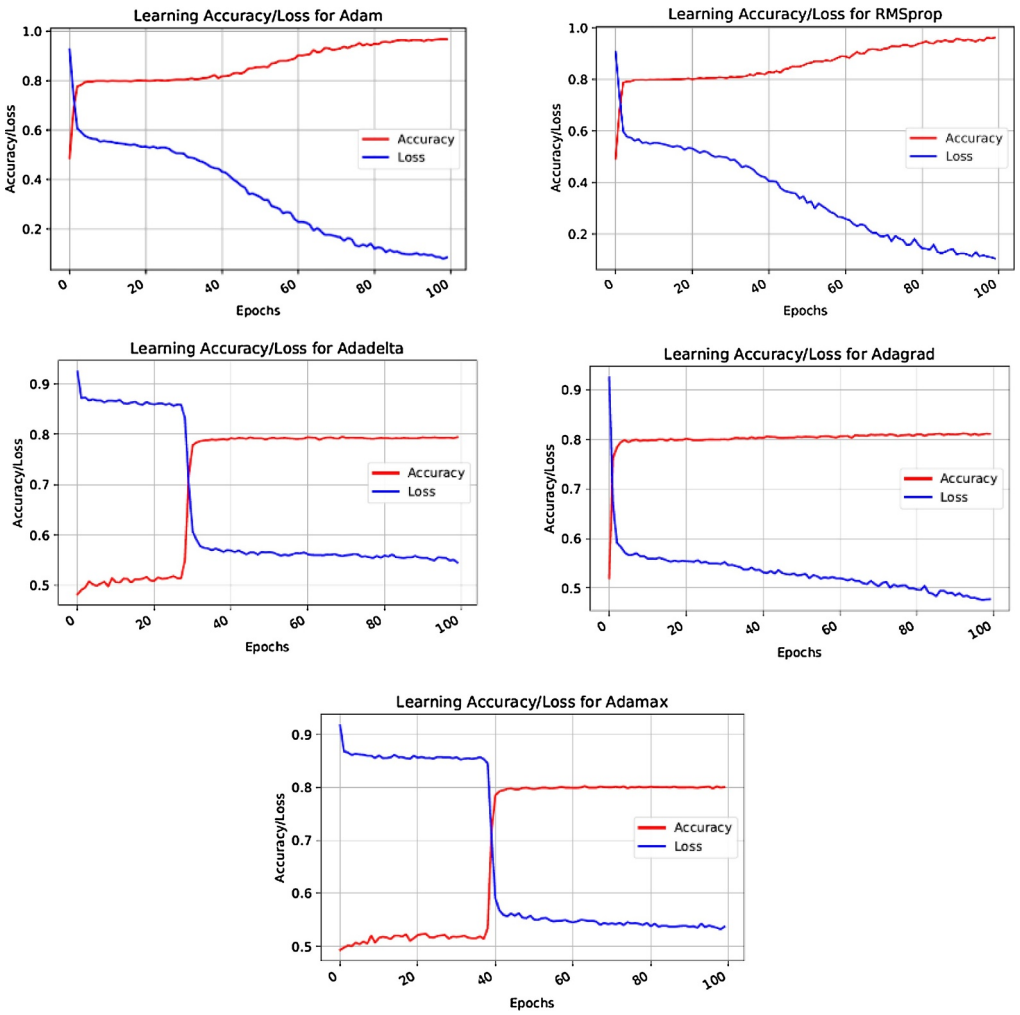

Fig. 6. Learning curve of the sentence classification network with different optimizers.

Table 5
Algorithmic metadata evaluation.

Methods	Optimizer	Precision	Recall	F1-score	Accuracy
Rule-based	N/A	0.211	0.163	0.155	0.501
SVM	N/A	0.701	0.622	0.562	0.624
DNN	RMSprop	0.787	**0.778**	**0.782**	**0.785**
DNN	Adam	**0.809**	0.777	0.781	0.755

4.4. Rule-based Method

For a baseline benchmark, we designed a keyword-based approach. In this technique, we carefully extracted the algorithmic metadata related keywords such as *dataset, corpus, running time, complexity, precision, f-score*, etc. from a subset of our dataset. Furthermore, we employed a simple keyword-matching algorithm to mark all matched lines as an algorithmic specific metadata text line. For instance, if a line contains any of the algorithmic specific keywords, it would be marked as the positive class line. The rest of the lines are treated as negative class instances. However, this traditional keyword matching approach suffered from very low recall and low precision due to its inability to capture the context of the sentences. Furthermore, we deployed an SVM model with linear kernel settings. Table 5 shows our achieved results in terms of precision, recall, F1-score, and accuracy for our proposed deep neural network (DNN) algorithm metadata sentence classification model, and the other two baselines (i.e. Rule-based and SVM methods). The deep learning model trained with the RMSprop optimizer yielded the highest F1 score. Therefore, we used these results to represent the best performance achieved by our proposed approach. Overall, our deep neural network approach performed better than both the Rule-based and SVM approaches, with 79% accuracy, due to its ability to understand representations of semantics and context during classification. It is also worth noting that the Bi-Directional LSTM treats an input sentence as a sequence of terms, whose orders are considered, while the other baselines represent a sentence as a bag of words. The high classification performance achieved by the Bi-Directional LSTM may infer that the orderings of terms could be useful signals for detecting sentences with algorithm-specific information in a scientific document.

4.5. Employed Search Systems

In this section, we present a prototype search system, designed specifically to target searching for documents containing algorithms. We generated document improved summary called *document synopsis* using algorithmic metadata extracted sentences and algorithm referencing text lines from the document using the BM25 formula [31]. Equation 4 represents the mathematical formulation for BM25.

$$BM25(Q,\ L) = \sum_{t \in q} \left\{ \log \frac{n}{lc_t} \times \frac{(k_1 + 1) tc_{tl}}{k_1 \left((1 - b) + b \times \left(\frac{l_L}{l_{av}} \right) \right) + tc_{tl}} \times \frac{(k_3 + 1)\ tc_{tq}}{k_3 + tc_{tq}} \right\} \tag{4}$$

Where n represents the number of lines in the document, lc_t = line count, i.e., the number of lines that contain the term t, tc_{ts} = number of times a term t appears in a line L, tc_{tq} = frequency of term t in query Q, l_L = length of line L, l_{av} = average length of a line in D and constants $k1 = 2$, $k3 = 2$, and $b = 0.75$.

The synopsis presents the algorithmic specific brief description of a research publication. Representing a document with its algorithmic synopsis facilitates the algorithm searchers to narrow down search results, especially when their information needs are algorithm related. Therefore, we used document synopses for indexing instead of a complete document.

In order to measure the usefulness of the synopsis generated by our proposed methods in terms of improving searching capabilities, we carried out an empirical study on the ACL dataset containing 21,940 documents. Firstly, we deployed our algorithmic PC metadata extraction model on those papers which have one or more algorithmic PCs. Secondly, we created a *document synopsis* using our proposed algorithmic PC metadata detection. Lastly, we indexed these synopses and calculated their ranking results (R) on over 50 algorithm related search queries (see Table 6). Additionally, we computed Reciprocal Rank (RR) by taking inverse of R, RR = 1/R and Mean RR (MRR) by taking sum average of RR, MMR = $\frac{\text{sum ofall RR}}{\#\ \text{of queries}}$. In this experiment, we implement a baseline search engine using Apache Lucene that indexes whole documents and use customized TF-IDF based textual similarity to retrieve documents. By design, Lucene implements a customized TF-IDF function for efficient score computation, using techniques for boosting and normalizing.

Fig. 7 illustrates the RR for both the traditional system (TS) and the enhanced system (ES) for the 50 queries. The box areas represent the RR value for TS and ES against 50. In the case of ES, we observed a higher RR score for most of the queries. Furthermore, Fig. 7 shows the MRR values for both systems. The red line represents the MRR value of the corresponding system. Our ES achieved average MMR = 0.7 against TS that yielded a lower average MRR of 0.49. Therefore, in the case of ES, the MRR line is towards the upper end of the graph compared to TS. Overall, we observed around 40% increases in the performance of our proposed ES compared to the TS in terms of mean reciprocal rank.

Table 6
Sample query ranking results for the advanced system and traditional system.

Queries	Advance System [1/R]	Traditonal System [1/R]	ACL paper ID	Paper Title
Q1: Graph Disambiguation Algorithm	1 [1]	10[0.1]	D11-1072	Robust Disambiguation of Named Entities in Text
Q2: Recursive Propagation algorithm for Propagated affinity matrix	1[1]	1[1]	D09-1091	Multilingual Spectral Clustering Using Document Similarity Propagation
Q3: Word Sense Subjectivity Score scoring algorithm on SENSIVAL-3	1[1]	3[0.33]	P06-1134	Word Sense and Subjectivity
Q4: BFS algorithm on TREC dataset having high precision and recall	1[1]	6[0.16]	D12-1138	A Discriminative Model for Query Spelling Correction with Latent Structural SVM
Q5: Averaged perceptron algorithm	2[0.5]	18[0.05]	D09-1161	K-Best Combination of Syntactic Parsers

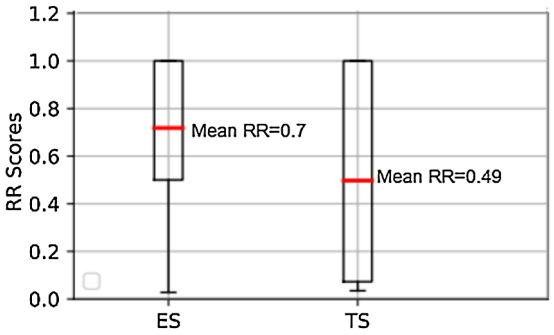

Fig. 7. Reciprocal Rank for both the Enhanced System (ES) and Traditional System (TS)

5. Concluding Remarks

Algorithms are ubiquitous in computer science and related fields. The ability to extract and index algorithms and their metadata from scientific documents could prove valuable for computational science researchers and software developers looking for cutting-edge algorithmic solutions to specific problems that need to be solved. Therefore, the ability and capacity to perform complex queries and arrive at highly accurate findings are of the benefit of the public and private sector, academia included. In this paper, we proposed a novel machine learning-based technique that – by reference to 60 distinctive features – enables the identification of algorithmic pseudo-codes present in scientific publications. Additionally, we identified algorithmic metadata sentences that while using deep neural networks enhance the searching competences of an algorithm focused IR system. In the future, we plan to design a new set of ETL pipelines to design character embeding based PC extraction techniques Since, the existing embedding based techniques cannot fulfill this task efficiently at their present form because they can only look into the given PC line and cannot grasp the contents of a PC box that are written in different font sizes, in multiple styles and spans over multiple lines, separated from the body of text in a scholarly article. Furthermore, we plan to explore NLP based techniques, such as tree parsing, etc. to understand the text semantics using alphanumeric characters (Batista-Navarro et al., 2013). This would allow us to compare algorithms based on their observed results on a particular data corpus with their achieved time complexity. Furthermore, we can design techniques for finding other algorithm metadata such as code snippets, input, output, and compatible data structures in a large-scale data corpus for improved document search (Nawaz, Thompson & Ananiadou 2012; Jahangir et al., 2017). Last but not the least, this study is helpful to build IR models for researchers, scientists, academics and practitioners working in applied sciences, who are inclined to find relevant knowledge related to algorithms from scientific publications. Therefore, we foresee the potential applications of this study in an emerging area of bibliometric-enhanced information retrieval. As mentioned above, this, in turn, would of particular interest in the public and private sector, not least in the process of information retrieval, comparison, match findings, etc. The data and code used in this paper can be downloaded from here: https://github.com/slab-itu/algo-metadata.

CRediT authorship contribution statement

Iqra Safder: Conceptualization, Methodology, Validation, Formal analysis, Software, Writing - original draft. **Saeed-Ul Hassan:** Supervision, Funding acquisition. **Anna Visvizi:** Supervision, Writing - review & editing. **Thanapon Noraset:** Supervision, Writing - review & editing. **Raheel Nawaz:** Supervision, Writing - review & editing. **Suppawong Tuarob:** Supervision, Conceptualization, Methodology, Investigation, Resources, Data curation, Formal analysis, Writing - review & editing.

Acknowledgments

This research is supported by the NRPU Grant # 6857, funded by the Higher Education Commission of Pakistan, and partially supported by the Thailand Science Research and Innovation (TSRI), formerly known as Thailand Research Fund (TRF), through grant RSA6280105.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ipm.2020.102269.

References

Al-Zaidy, R. A., & Giles, C. L. (2017). A machine learning approach for semantic structuring of scientific charts in scholarly documents. *Twenty-Ninth IAAI Conference*.
Al-Zaidy, R. A., & Giles, C. L. (2018). Extracting semantic relations for scholarly knowledge base construction. *In 2018 IEEE 12th international conference on semantic computing (ICSC)* (pp. 56–63). IEEE.
Altınel, B., & Ganiz, M. C. (2018). Semantic text classification: A survey of past and recent advances. *Information Processing & Management, 54*(6), 1129–1153.
Ananiadou, S., Thompson, P., & Nawaz, R. (2013). Enhancing search: Events and their discourse context. *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 318–334). Berlin, Heidelberg: Springer.
Arshad, N., Bakar, A., Soroya, S., Safder, I., Haider, S., Hassan, S., Aljohani, N., Alelyani, S., & Nawaz, R. (2019). *Extracting scientific trends by mining topics from Call for Papers.* Library Hi Tech, in presshttps://doi.org/10.1108/LHT-02-2019-0048.
Azad, H. K., & Deepak, A. (2019). Query expansion techniques for information retrieval: a survey. *Information Processing & Management, 56*(5), 1698–1735.
Batista-Navarro, R. T., Kontonatsios, G., Mihăilă, C., Thompson, P., Rak, R., Nawaz, R., Korkontzelos, L., & Ananiadou, S. (2013). Facilitating the analysis of discourse phenomena in an interoperable NLP platform. *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 559–571). Springer.
Bakar, A., Safder, I., & Hassan, S.-U. (2018). Mining algorithmic complexity in full-text scholarly documents. *ICADL Poster Proceedings*The University of Waikato.
Bhatia, S., & Mitra, P. (2012). Summarizing figures, tables, and algorithms in scientific publications to augment search results. *ACM Transactions on Information Systems (TOIS), 30*(1), 3.
Choudhury, S. R., Wang, S., & Giles, C. L. (2016). Curve separation for line graphs in scholarly documents. *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries* (pp. 277–278). .
Clark, C., & Divvala, S. (2016). PDFFigures 2.0: Mining figures from research papers, in: Digital Libraries (JCDL). *2016 IEEE/ACM Joint Conference On* (pp. 143–152). IEEE.
Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very deep convolutional networks for text classification. *ECACL, 1*, 1107–1116.
Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems, 28*(10), 2222–2232.
Hassan, T. (2009). Object-level document analysis of PDF files. *Proceedings of the 9th ACM symposium on Document engineering* (pp. 47–55). ACM.
Huang, M., Qian, Q., & Zhu, X. (2017). Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS), 35*(3), 26.
Imran, M, Akhtar, A, Said, A, Safder, I, Hassan, SU, & Aljohani, NR (2018). Exploiting social networks of Twitter in altmetrics big data. *In23rd international conference on science and technology indicators (STI 2018)*Centre for Science and Technology Studies (CWTS) September 12-142018 Sep 11.
Jahangir, M., Afzal, H., Ahmed, M., Khurshid, K., & Nawaz, R. (2017). An expert system for diabetes prediction using auto tuned multi-layer perceptron. *2017 Intelligent Systems Conference (IntelliSys)* (pp. 722–728). IEEE.
Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *European conference on machine learning* (pp. 137–142). Springer.
Jorge, A., & de Andrade Lopes, A. (1999). June *Iterative part-of-speech tagging. In International Conference on Learning Language in Logic* (pp. 170–183). Springer.
Khabsa, M., Treeratpituk, P., & Giles, C. L. (2012). Ackseer: a repository and search engine for automatically extracted acknowledgments from digital libraries. *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries* (pp. 185–194). ACM.
Kim, Y. (2014). Convolutional neural networks for sentence classification. *EMNLP, 2014* 1746–1175.
Kim, Y, Jernite, Y, Sontag, D, & Rush, AM (2016). Character-aware neural language models. *InThirtieth AAAI Conference on Artificial Intelligence* Mar 5.
Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Twenty-ninth AAAI conference on artificial intelligence*.
Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *In International conference on machine learning* (pp. 1188–1196). .
Li, X., & de Rijke, M. (2019). Characterizing and predicting downloads in academic search. *Information Processing & Management, 56*(3), 394–407.
Mai, F., Galke, L., & Scherp, A. (2018). Using Deep Learning for Title-Based Semantic Subject Indexing to Reach Competitive Performance to Full-Text. *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries* (pp. 169–178). ACM.
Meuschke, N., Gondek, C., Seebacher, D., Breitinger, C., Keim, D., & Gipp, B. (2018). An adaptive image-based plagiarism detection approach. *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries* (pp. 131–140). ACM.
Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* (pp. 3111–3119). .
Milidiú, R. L., Laber, E. S., & Pessoa, A. A. (1999). A work-efficient parallel algorithm for constructing Huffman codes. *Proceedings DCC'99 Data Compression Conference* (pp. 277–286). IEEE Cat. No. PR00096.
Mitra, P, Giles, C. L., Sun, B., & Liu, Y. (2007). Chemxseer: a digital library and data repository for chemical kinetics. *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience* (pp. 7–10). ACM.
Mahmood, Z., Safder, I., Nawab, R. M. A., Bukhari, F., Nawaz, R., Alfakeeh, A. S., ..., & Hassan, S. U. (2020). Deep sentiments in Roman Urdu text using Recurrent Convolutional Neural Network model. *Information Processing & Management, 57*(4), 102233.
Nawaz, R., Thompson, P., & Ananiadou, S. (2012). Identification of Manner in Bio-Events. *LREC* (pp. 3505–3510). .
Nguyen, H. T., Duong, P. H., & Cambria, E. (2019). Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowledge-Based Systems*104842.
Petrakis, E., & Georgiadis, C. (2000). Evaluation of spatial similarity methods for image retrieval. *Conference on Signal Processing Communications and Computer Science* (pp. 13–18). .
Rastan, R., Paik, H. Y., & Shepherd, J. (2019). TEXUS: A unified framework for extracting and understanding tables in PDF documents. *Information Processing & Management, 56*(3), 895–918.
Rubin, T. N., Chambers, A., Smyth, P., & Steyvers, M. (2012). Statistical topic models for multi-label document classification. *Machine learning, 88*(1-2), 157–208.
Rush, A. M., Harvard, S. E. A. S., Chopra, S., & Weston, J. (2017). A neural attention model for sentence summarization. *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*.
Safder, I., & Hassan, S. U. (2019). Bibliometric-enhanced information retrieval: a novel deep feature engineering approach for algorithm searching from full-text publications. *Scientometrics, 119*(1), 257–277.
Safder, I., & Hassan, S. U. (2018). DS4A: Deep search system for algorithms from full-text scholarly big data. *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 1308–1315). IEEE.

Safder, I., Hassan, S. U., & 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. *Companion Proceedings of the The Web Conference 2018* (pp. 251–258). International World Wide Web Conferences Steering Committee.

Safder, I., Sarfraz, J., Hassan, S. U., Ali, M., & Tuarob, S. (2017). Detecting target text related to algorithmic efficiency in scholarly big data using recurrent convolutional neural network model. *International conference on Asian digital libraries* (pp. 30–40). Springer.

Hassan, SU, Safder, I, Akram, A, & Kamiran, F (2018). A novel machine-learning approach to measuring scientific knowledge flows using citation context analysis. *Scientometrics, 116*(2), 973–996 Au g 1.

Hassan, SU, Imran, M, Iftikhar, T, Safder, I, & Shabbir, M (2017). *Deep stylometry and lexical & syntactic features based author attribution on PLoS digital repository. InInternational conference on Asian digital libraries*. Springer119–127 Nov 13.

Shardlow, M., Batista-Navarro, R., Thompson, P., Nawaz, R., McNaught, J., & Ananiadou, S. (2018). Identification of research hypotheses and new knowledge from scientific literature. *BMC medical informatics and decision making, 18*(1), 46.

Siegel, N., Lourie, N., Power, R., & Ammar, W. (2018). Extracting scientific figures with distantly supervised neural networks. *Proceedings of the 18th ACM/IEEE on joint conference on digital libraries* (pp. 223–232). ACM.

Siegel, N., Horvitz, Z., Levin, R., Divvala, S., Farhadi, A., 2016. FigureSeer: Parsing result-figures in research papers, in: European Conference on Computer Vision. Springer, pp. 664–680.

Sinoara, R. A., Camacho-Collados, J., Rossi, R. G., Navigli, R., & Rezende, S. O. (2019). Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems, 163*, 955–971.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. *Thirteenth annual conference of the international speech communication association*.

Suzuki, T., & Fujii, A. (2017). Mathematical document categorization with structure of mathematical expressions. *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1–10). IEEE.

Tuarob, S., Bhatia, S., Mitra, P., & Giles, C. L. (2016). AlgorithmSeer: A system for extracting and searching for algorithms in scholarly big data. *IEEE Transactions on Big Data, 2*(1), 3–17.

Tuarob, S., Mitra, P., & Giles, C. L. (2015). A hybrid approach to discover semantic hierarchical sections in scholarly documents. *2015 13th international conference on document analysis and recognition (ICDAR)* (pp. 1081–1085). IEEE.

Vo, D. T., & Bagheri, E. (2019). Feature-enriched matrix factorization for relation extraction. *Information Processing & Management, 56*(3), 424–444.

Wang, X., Rak, R., Restificar, A., Nobata, C., Rupp, C. J., Batista-Navarro, R. T. B., ..., & Ananiadou, S. (2011). Detecting experimental techniques and selecting relevant documents for protein-protein interactions from biomedical literature. *BMC bioinformatics, 12*(8), S11.

Xiong, C., Power, R., & Callan, J. (2017). Explicit semantic ranking for academic search via knowledge graph embedding. *Proceedings of the 26th international conference on world wide web* (pp. 1271–1279). International World Wide Web Conferences Steering Committee.

Xia, F., Wang, W., Bekele, T. M., & Liu, H. (2017). Big scholarly data: A survey. *IEEE Transactions on Big Data, 3*(1), 18–35.

Xie, I., Babu, R., Lee, T. H., Castillo, M. D., You, S., & Hanlon, A. M. (2019). Enhancing usability of digital libraries: Designing help features to support blind and visually impaired users. *Information Processing & Management*102110.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems* (pp. 649–657). .

Zhang, X., & LeCun, Y. (2015). Text understanding from scratch. arXiv preprint arXiv:1502.01710.