Assembling appropriate information with help of different Data Mining Technique

[1] Princy Mishra, [2,] Ritu Powar [3] Vaishnavi Sonawane [4] Prof. A. D. Khairkar

Abstract: Computer science and many of its applications are about developing, analyzing, and applying algorithms. Efficient solutions to important problems in various disciplines other than computer science usually involve transforming the problems into algorithmic ones on which standard algorithms are applied. Scholarly Digital documents are increasing day by day. To automatically find and extract these algorithms in this vast collection of documents that enable algorithm indexing, searching discovery, and analysis. AlgorithmSeer, a search engine for algorithms, has been investigated as part of CiteSeerX with the intent of providing a large algorithm database. A novel set of scalable techniques used by AlgorithmSeer to identify and extract algorithm representations in a heterogeneous pool of scholarly documents is proposed. Along with this, anyone with different levels of knowledge can access the platform and highlight portions of textual content which are particularly important and relevant. The highlighted documents can be shared with others in support of lectures and self-learning. But the highlighted part of the text cannot be useful to different levels of learners. We solve the problem of predicting new highlights of partly highlighted e-learning documents.

IndexTerms **- Component,formatting,style,styling,insert.**

I. INTRODUCTION

Computer science is about developing, analyzing, and applying algorithms. Efficient solutions to important problems in various disciplines other than computer science usually involve transforming the problems into algorithmic ones on which standard algorithms are applied. Furthermore, a thorough knowledge of state-of-the-art algorithms is also crucial for developing efficient software systems. Standard algorithms are usually collected and cataloged manually in algorithm textbooks, encyclopedias, and websites that provide references for computer programmers. While most standard algorithms are already cataloged and made searchable, especially those in online catalogs, newly published algorithms only appear in new articles. The explosion of newly developed algorithms in scientific and technical documents makes it infeasible to manually catalog these newly developed algorithms. Manually searching for these newly published algorithms is a nontrivial task. Researchers and others who aim to discover efficient and innovative algorithms would have to actively search and monitor relevant new publications in their fields of study to keep abreast of the latest algorithmic developments. The problem is worse for algorithm searchers who are inexperienced in document search.

We would like to have a system that automatically discovers and extracts algorithms from scholarly digital documents. Such a system could prove to facilitate algorithm indexing, searching, and a wide range of potential knowledge discovery applications and studies of the algorithm evolution, and presumably increase the productivity of scientists. Since algorithms represented in documents do not conform to specific styles and are written in arbitrary formats, this becomes a challenge for effective identification and extraction.

Keyword:

AlgorithmSeer, CiteSeerX, scholarly document.

Related Work:

As like our system CiteSeer, TableSeer also use the searching algorithm but it has the following drawbacks
CiteSeer:
Though useful algorithms could be found, the search results can be contaminated with irrelevant items.

TableSeer:
It only provides the tables format data to the user. No extra information is given with algorithms.

Motivation:

Computer science and many of its applications are about developing, analyzing, and applying algorithms.

There is need for automated systems that efficiently identify, extract, index and search this ever increasing collection of algorithmic document

It is infeasible to manually catalog newly developed algorithms

For searchers who are inexperienced in document search find it difficult to search related algorithms.

Mathematical Model

S = {s, e, X, Y, F, A}

S= Set Theory

s = Start of the program

1. Register/Login into the system

e = End of the program

X = input of the program = PDF Document

Y = Output of program = Search for Content's

A=Success of program= 1.PDF Document

2. Get the Content's

3. Result of PDF

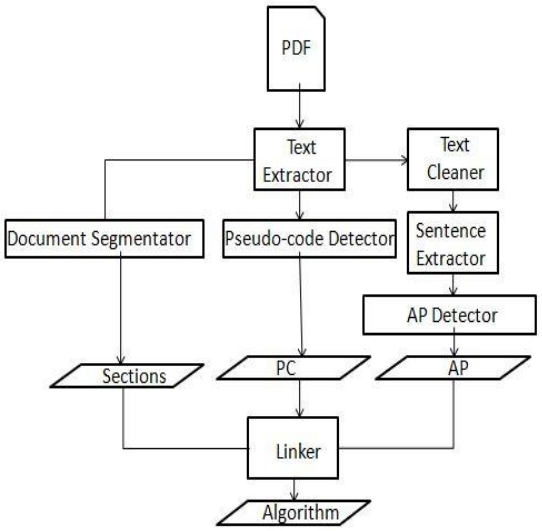
F = Failure of Program= PDF Not Found.

First, user provide PDF document from Admin.
Let F be the set of features
F= {F1, F2, ..., Fn}
Admin will Upload the PDF document as blob file and system will divide into three tables.
User can search specific PDF and it will display all the information of PDF like (Abstract, Algorithm, Highlighted words

$D= Sq.rt((F_{11}-F_{12})^2+(F_{21}-F_{22})^2+...+(F_{n1}-F_{n2})^2)$
Fixed n number of PDF is then extracted. Label for each PDF is then checked. The PDF will divide into three tables.
Search for Specific PDF it will display PDF on the basis of TFIDF

System Architecture:

Figure shows the detailed flow of our System. In this user can upload PDF and algorithm name as an input. Using core NLP technique and Content-based filtering, given PDF file or our PDFs in our database will be processed. We are going to perform an operation like stemming, stop words removal and parsing technique. After this semantic similarity check will be performed on a word or concept level as well as document or text level. Based on the similarity check TFIDF values will be calculated from words present in an already uploaded document. Based on TFIDF values plagiarism report will be returned to the user in the form of duplicate content and graphical representation.



System Architecture

To compute the TF-IDF value for a one document, we take that document and calculate the *TF-IDF score* for each *unique* word without stop words. For each unique word, or *term*,

1. Term frequency: this is used for measuring how frequently particular term appears in documents.

$$TF(term, document) = \frac{\text{Number of times the term appears in doc}}{\text{Total number of words in a doc}}$$

…… eq1

2. Inverse Document Frequency: this is used for measuring how frequently particular term appears in all documents.

$$IDF(term) = \begin{cases} 0 & \text{if term doesn't appear in any doc} \\ \ln\left(\frac{\text{Total number of docs}}{\text{Number of docs containing the term}}\right) & \text{Otherwise} \end{cases}$$

…eq2

3. On equation 1 and 2 calculate overall TFIDF of word,

TF-IDF= TF * IDF

Conclusion:

Algorithmseer an efficient system for searching algorithms from the huge collection of files (PDF documents) is proposed. The system uses core NLP techniques to extract the algorithmic information from the file. Also, the system is able to identify the highlighted content and keep them as short notes for that particular file or topic. This information can be used for improving the efficiency of searching techniques when it comes to searching algorithms only.

ACKNOWLEDGMENT:

It gives us great pleasure in presenting the preliminary project report on **'Assembling appropriateinformation with help of different Data Mining Technique '**

I would like to take this opportunity to thank my internal guide for giving me all the help and guidance I needed I am really grateful to them for their kind support. Their valuable suggestions were very helpful.
I am also grateful to HOD for her in dispensable support and suggestions.

Name of Students

[1] Princy Mishra, [2] Ritu Powar,[3] Vaishnavi Sonawane

REFERENCES

- 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 Proceedings of the 23rd national conference on Artificial intelligence - Volume 2, AAAI'08, pages 1169–1174. AAAI Press, 2008.
- J. B. Baker, A. P. Sexton, V. Sorge, and M. Suzuki. Comparing approaches to mathematical document analysis from pdf. ICDAR '11, pages 463–467, 2011.
- S. Bhatia, P. Mitra, and C. L. Giles. Finding algorithms in scientific articles. WWW '10, pages 1061–1062, 2010.
- D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, Mar. 2003.
- J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell., 20(3):226–239, Mar. 1998.
- S. Bhatia, S. Tuarob, P. Mitra, and C. L. Giles. An Algorithm Search Engine for Software Developers. 2011.

- TA. Asuncion, M. Welling, P. Smyth, and Y. W. Teh. On smoothing and inference for topic models. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, pages 27–34, Arlington, Virginia, United States, 2009. AUAI Press.
- CP. Chiu, F. Chen, and L. Denoue. Picture detection in document page images. DocEng '10, pages 211–214, 2010.
- S. Bhatia and P. Mitra. Summarizing figures, tables, and algorithms in scientific publications to augment search results. ACM Trans. Inf. Syst., 30(1):3:1–3:24, Mar. 2012.