Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm

Charles Adjetey[1]
Department Computer Science
Lancaster University Ghana
Accra, Ghana

Kofi Sarpong Adu-Manu[2]
Department Computer Science
University of Ghana
Legon-Accra, Ghana

***Abstract*—Image Retrieval Systems (IRSs) are applications that allow one to retrieve images saved at any location on a network. Most IRSs make use of reverse lookup to find images stored on the network based on image properties such as size, filename, title, color, texture, shape, and description. This paper provides a technique for obtaining full image document given that the user has some portions of the document under search. To demonstrate the reliability of the proposed technique, we designed a system to implement the algorithm. A combination of Optical Character Recognition (OCR) engine and an improved text-matching algorithm was used in the system implementation. The Tesseract OCR engine and Levenshtein Algorithm was integrated to perform the image search. The extracted text is compared to the text stored in the database. For example, a query result is returned when a significant ratio of 0.15 and above is obtained. The results showed a 100% successful retrieval of the appropriate file base on the match even when partial query images were submitted.**

Keywords*—Image Retrieval Systems; image processing; Optical Character Recognition (OCR), text matching algorithm, Tesseract OCR engine, Levenshtein Algorithm

I. Introduction

Advances in internet technology, creativity, innovation and the tremendous increase in digital image creation in documents have given rise to the use of techniques to detect and query images on the web. **Content-based image retrieval systems are well studied and a number of researchers have proposed a number of techniques [1], [2], [3], [4]. Some of the authors proposed a new computational backend model that includes a dataset and OCR services for Arabic document information retrieval (ADIR)[1], multimodal features for similarity search we apply re-ranking according to averaged or maximum scores [2] and diagram image retrieval and analysis, with an emphasis on methods using content-based image retrieval (CBIR), textures, shapes, topology and geometry [5].**

The use of sophisticated computer software and multimedia tools on the market has led to the production of digital information and digital images stored in different repositories and cloud servers around the globe. The web content may be searched using keywords or images to retrieve relevant documents or images using search engines. There are general-purpose search engines used for text-based information collections and specialized search engines that allow for special image collections on the web [6].

Search Engines (SEs) give users indexed lists that provide essential information on high-quality websites [7]. Search Engine (SE) may be explained as a program that scans for and recognises items in a database that relates to keywords or characters determined by the users [8]. SEs are utilized mainly to find specific sites on the World Wide Web (WWW) that contain relevant information of interest after querying databases that contain or potentially matching the user keywords. The WWW contains an immense amount of material that require search engines to act as filters to retrieve information on the web. SEs make it easier and serves as a quicker means for users to discover information significant to them without wading through the copious number of insignificant web pages [9].

Different kinds of SEs require the user to input text which serves as the keyword for the search to take place. Searching for specific information via text on the web is a daunting task because users keywords are expected to match the indexing data structure to gain relevant feedback or information. Apart from searching the WWW for information using text, searching via images have also gained more prominence over the past two decades using image search engines (ISEs). ISEs require an image as the input item for the search to take place.

There is a lot of documents and files with multimedia content (that is text and images) of which users seek to retrieve using portions of the file, similar files or other documents [10], [11]. Recently, researchers have aimed at designing algorithms that support image querying that is capable of retrieving images, documents containing images and extracting images within vast scale databases. Content-based image retrieval systems (CBIRS) are required to retrieve specific image descriptors from digital libraries efficiently. CBIRS may operate based on low-level or high-level semantics [9]. The approaches used in CBIRS include deep learning methods, Scale Invariant Feature Transform (SIFT) algorithm, Clustered-scale Invariant (CSI) algorithm, deep learning and compressed domain, convolutional neural networks (CNN), Dot-Diffused Block Truncation Coding (DDBTC), and the Relevance Feedback (RF) Model [12]. All these methods take user input, processes and return the retrieval results to the user. It is expected that the retrieval systems provide the exact results that are semantically similar to the user query.

Most CBIRS use visual content to identify relevant images. In these systems, three key issues arise: how the image is represented, organized, and the image similarity measurements. The general framework of CBIRS may be classified into two stages, the offline and the online stage [6].

Deep learning techniques such as deep belief network are recent approaches used for feature extraction in CBIR applications. In this paper, we based on the strength of the Tesseract OCR to recognize text (that is, makes documents editable and searchable) in an image and we then use our novel algorithm to extract the text from the image. After extracting the text from the image document, we apply the Levenshtein text-matching algorithm to find the similarity between the texts extracted from the images saved in the database. The Levenshtein serves as an edit-distance algorithm to provide information about the least number of edits needed to modify one string to obtain another. For example, a distance of zero (0) indicates that the two strings are identical anything else means they are not the same. The more prominent the distance, the more extraordinary the strings.

Many factors influence search results; among these factors, the quality of keywords depends on the users. Most users enter short queries and avoid using anything other than the primary search engine index; when the search results are returned, users typically would focus on the first few results that appear in the search results. They are primarily unlikely to modify the query. With the CBIR, the search query is not dependent on the user instead the features in the image hence eliminating biases that will lead to poor search.

In this paper, we focus on reverse image search engines (that is, query by example). There are two main techniques used when searching for images in a reverse image search engines: 1) searching with keywords, and 2) searching with pictures/images. In this paper, we search by images. A resulting document-image is retrieved from the database when there is a similarity with the search image. The uniqueness of our approach is that any portion of the image is capable of retrieving the full image-document from the database.

II. Related Work

Content-based image retrieval is gaining more attention in the research community and has been one of the growing areas of research in the past decade. In this section, we provide previous works and approaches related to content-based image retrieval systems. These approaches have used different metrics to retrieve contents of documents from repositories across the world. Over the years, several low-level feature descriptors proposed for image representation have depended on extracting underlying features such as colour, position, edge, shape, texture, GIST, CENTRIST, and bag-of-words (BoW) models using local feature descriptors such as SIFT and SURF. In what follows, we provide brief descriptions of previous works done in the CBIR systems.

In [12], the authors proposed a multi-feature image retrieval technique that was capable of combining features such as colour, edge, and texture to retrieve an image. They made use of a deep belief network (DBN) of Deep Learning to extract these features from an image. The retrieval performance of their approach depended on the feature representation and similarity measurement. In their approach, the DBNs applied in CBIR proposed by authors aided in understanding learning and mapped the various features extracted from the images provided by users. A thorough investigation of feature extraction, mapping, zero paddings, residual learning and loss function were discussed. Their results showed that for a data set of 1000 images, an accuracy rate of about 98.6% was obtained during real-time data extraction. The authors in [13], [12], [11] designed a web content-based image retrieval system referred to as Anaktisi. The search for images is based on various descriptors which include colour and texture. Input from the user can be any image or keyword. Based on the input, the search is conducted in the various databases. The user is presented with images similar to the search input. The presentation of the images is base on the following descriptors: colour and edge directivity descriptor, the fuzzy colour and texture histogram, join composite descriptor, and their compact variants. The authors recommended that future work should include the capacity to seek document images by word spotting.

In [14], the authors developed an interactive content retrieval system. The system had a Graphical User Interface (GUI) that use colour, texture, and shape of a given image and performs a similarity search of the image. The application is made up of three sections. The image laboratory section presents the low-level features of the image used by the application for the retrieval procedure. The image retrieval section makes use of three different search options to retrieve the image, and the extras section of the system incorporates other application to check for the existence of an image. The searches of the images did not consider text that may be embedded in the query images in this system.

In [15], [16], the authors designed an automated system for search and retrieval of trademarks. In their work, they introduce a software that will enable the searching, comparing and retrieving of trademarks. The software comprised of: (i) annotation and storing trademarks in a database; (ii) text-based search and retrieval of trademarks based on annotations; and (iii) content-based search and retrieval based on trademark images. The system was not capable of recognising different trademark features due to the inefficiency of the algorithm used. The system's matching accuracy of trademark images was low and was not capable of assessing real-time data store containing trademarks. In their system, they did not specifically address how their system would deal with the text that is embedded in the word-in-mark trademarks.

The authors in [16] designed a mobile visual search using images and text features. In their work, they propose a mobile visual search system that uses both text and low-bit image features. They described how a phone enabled with a camera can be used to take a photo of a document image and search for the document in online databases. In the detection of text, the authors make use of OCR to detect the text. After the detection of the text, the system identifies the title text. The title text is the one identified because the most informative keywords to search for a document is in the title; hence it is extracted. Based on the features that are characterised by text titles such as size and stroke width, they are identified. Before the extracted features are passed to the OCR, they are rectified, to improve the accuracy of the features. Even though the query image is used as the source of the query and the focus is on the embedded text, it only considers the title text of documents for obvious reasons, and this may not be good enough in situations where the document title may not be available.

In [17], the authors proposed techniques to extract image descriptors using scale-invariant feature transform (SIFT) after which *k-means* clustering is performed on the features extracted with SIFT. These techniques were adopted due to their efficiency in retrieving realistic images in large databases. Experimental results, as reported by authors, showed that the proposed approach was efficient. The proposed approach when compared with the traditional Bag of Words (BoW) approach and the Spatial pyramid matching (SPM) yielded 92.69% mean Average Precision (mAP) whereas BoW and SPM achieved 82.00%and 88.80% mAP, respectively. In [18], the authors used 1400 binary and gray-scale images grouped into 70 categories with 20 images in each of the 70 categories were obtained from MPEG-7, COIL-20 and ZuBuD image databases. The authors observed that for both binary and gray-scale images, similar objects were retrieved. SIFT was used to perform the feature extraction to detect key-points within the datasets. The key-point descriptors were initially computed using the gradient magnitude and the orientation of the image at each image sample point within the chosen region. For example, the authors computed a 2X2 descriptor from a 4X4 set and 8X8 descriptor from a 16X16 image set samples. In the experiment results, the precision and recall graph of images such as octopus, hammer, apple and spoon revealed that images of octopus outperformed the remaining images due to the number of corners observed in an octopus. Hence, the SIFT algorithm was seen to perform better with images with more corners.

In the work of [9], the author designed an experiment and used datasets from the SUN database. The author classified the images into eight classes (that is, water, car, mountain, ground, tree, building, snow, sky and unknown - containing all other classes). Three thousand (3000) images placed into 41 categories were collected from the database. The images were further categorized into training images (1900), testing images (600) and validation images (500). Convolutional neural network (CNN) was employed for extracting required features. The CNN was repeatedly applied as a function across the sub-regions of the entire image. The focus was the input image with a linear filter, with an added bias term and further applying a non-linear function. The CNN was trained by passing the compressed train, test, and validation datasets. The system design comprised of the query image, trained neural network, and annotation index to obtain the results. From the authors evaluation results, it turned out that CNN outperformed Bag-of-Words (BoWs) in retrieving the different images (that is, water, car, mountain, ground, tree, building, snow, sky) but BoWs recorded fewer error rates in identifying a tree, building and sky. In Table I, a comparison of some existing content-based retrieval systems are provided. The comparison highlighted the techniques used for feature extraction, the query technique used, the testing environment, and the features used for the extraction.

From the literature review analysis, we observed that most of the research works in CBIR have focused on extracting images from database repositories using different feature extraction techniques. Besides the query by example scheme adopted in most projects, other query schemes such as querying by keyword and in some instances querying by colour layout have been used in recent implementations. Existing CBIR systems use the query by example scheme to retrieve similar images from databases. Most of these CBIR systems focused on retrieving images that contain objects from databases. In our work, we focused on the text feature of the query by example to retrieve text-based documents converted to image using part of the image as the query to retrieve the full documentation that contains the said text. This novel approach is relevant in searching large database repositories that contain an enormous amount of scanned text documents.

III. Methodology

A. Methods and Techniques

The system analysis and design tools used to model the system included Data Flow Diagram (DFD) and Unified Modelling Language (UML). Also, the incremental development and the reuse-oriented software engineering models were adopted for the development of the system. The main philosophy of this process is to develop an initial implementation, get feedback from the user and evolve the software until there is satisfaction with the functionalities of the software. Reuse-oriented software engineering approach was adopted because of its significant number of reusable components. The choice of the OCR engine influenced the usage of the model. The OCR engine adopted is the Tesseract-OCR. The use of the OCR engine decreases the amount of software to be developed. Also, it decreases the cost and risk that comes with using this approach [23].

Algorithm 1 Administrator Algorithm

```
Input: UPLOAD Image files
Output: Image
1: NAME Uploaded files
2: CALLS OCR module to extract text
3: SAVE extracted text to Database
4: SAVE Uploaded files (under the given Name) to a file
   location
5: return Image
```

B. Conceptual Model

Fig. 1 describes the conceptual framework for system implementation. The framework is in three compartments: the Query, Processing and Output, and Load. This framework guided the implementation of the proposed image retrieval system. The administrator activities are shown in Algorithm 1, and the user activities are also shown using Algorithm 2. In what follows, we provide a detail description of the activities involved in image retrieval. The system operates based on the following algorithms. The entire process flow is shown in Fig. 2 and Fig. 3.

C. Tools

The Tesseract OCR engine was used as the OCR engine for the system. The version used in our development supports multiple languages. It supports multiple image formats supported by Leptonica and supports layout analysis [24], [21]. ASP.NET MVC framework with C# framework was used for the development. Thus, the application does not need any installation on client computer beyond the usage of a web browser; making the application hardware independent

TABLE I. COMPARATIVE STUDY OF EXISTING CBRIS

Reference	CBIR Technique used for extraction	Brief Description	Query Techniques used and features extracted	Testing Environment
[12]	Deep belief network (DBN) method of deep learning	Proposed an efficient algorithm to search large databases and retrieve digital images using the feature representation and similarity measurements.	Query by example/colour, radiance, luminance, structures, frequency	Simulation [the type of simulator was not stated though by the authors]
[17]	Scale-invariant feature transform (SIFT)	The CBIR techniques introduced by the authors overcame the memory usage and matching time problem in SIFT. k-means clustering was employed to cluster the entire matrix into 16-clusters (that is a matrix of size 8X16). After this, a second approach is used to further reduced size 1 X 18	Query by example/Image scale, noise and illumination	Experimental testbed [Caltech 101 (with 9144 images) and Li database (with 2360) images]
[18]	Scale Invariant Feature Transform (SIFT) algorithm for binary and gray scale images	SIFT was employed to detect images and extract features of the image.	Query by example/Corners, edges, scale, rotation and translation	Experimental (MATLAB)
[9]	Convolutional Neural Networks (CNN)	The worked employed CNN to retrieve images from a database containing 3000 images that were placed into 41 categories and 8 classes	Query by example/colour and texture	Experimental
[19]	Convolutional Neural Networks (CNN) model and Dot-Diffused Block Truncation Coding (DDBTC)	In this work, two metrics (i.e., average precision rate and recall rate) to examine datasets to measure the retrieval rate. They employed deep learning with compression for image retrieval.	Query by colour and Example/Texture and colour	Experimental
[20]	Not provided	The work mainly focused on identifying problems existing in CBIR systems, biometric systems, and problems related to image feature extraction	Query by example/Shape, colour and texture	Investigation
[21]	Optical Character Recognition (OCR).	The work sought to improve OCR quality of digitizing historical collections in a national library in Finland. To show the improvement of their approach to other approaches, they employed morphological analyzer and character accuracy rate	ABBYY FineReader v.11 and Tesseract re-OCR	Evaluation
[22]	Relevance Feedback (RF) Model/ Deep Learning	The authors used the Rocchio algorithm to obtain the image vector trained in CNN model in a sorted database based on a similarity metric. The results obtained from an experimented design showed that the Precision-Recall results outperforms them on the Caltech256 datasets with the distance learning metric.	Empirical Studies/ Experimental	Experimental

Algorithm 2 Searcher Algorithm

```
Input: Query Image File
Output: Image
   Initialisation:
1: CALL OCR module to extract text
   LOOP Process
2: SAVE extracted text to temporary location
3: CALL Matching module
4: if (Match found) then
5:     DISPLAY Results of Matches
6:     RETRIEVE Document associated with Match from File
       location
7: end if
8: return $Image$
```

and ubiquitous. ASP.NET MVC allows the creation of web application by dividing the application into the model (M), views (V), and controllers (C) and this reduces the complexity of managing the broader application. The MVC framework provides the need for the separation of the business logic layer of the web application from its presentation layer. Unlike the Web Form framework, the MVC allows one to have absolute control on HTML and HTTP. MVC is extensible, providing developers with the ability to customize the framework [25].

1) Searcher Query: This is the part that interfaces with the user. The Searcher submits the query image file via the GUI, as shown in Fig. 4. The image file is sent to the processing and output section. The Query part serves as the front end of the system. The Searcher then searches for a particular document and uploads the image into the system. After the necessary processing has been performed on the image, a notification is

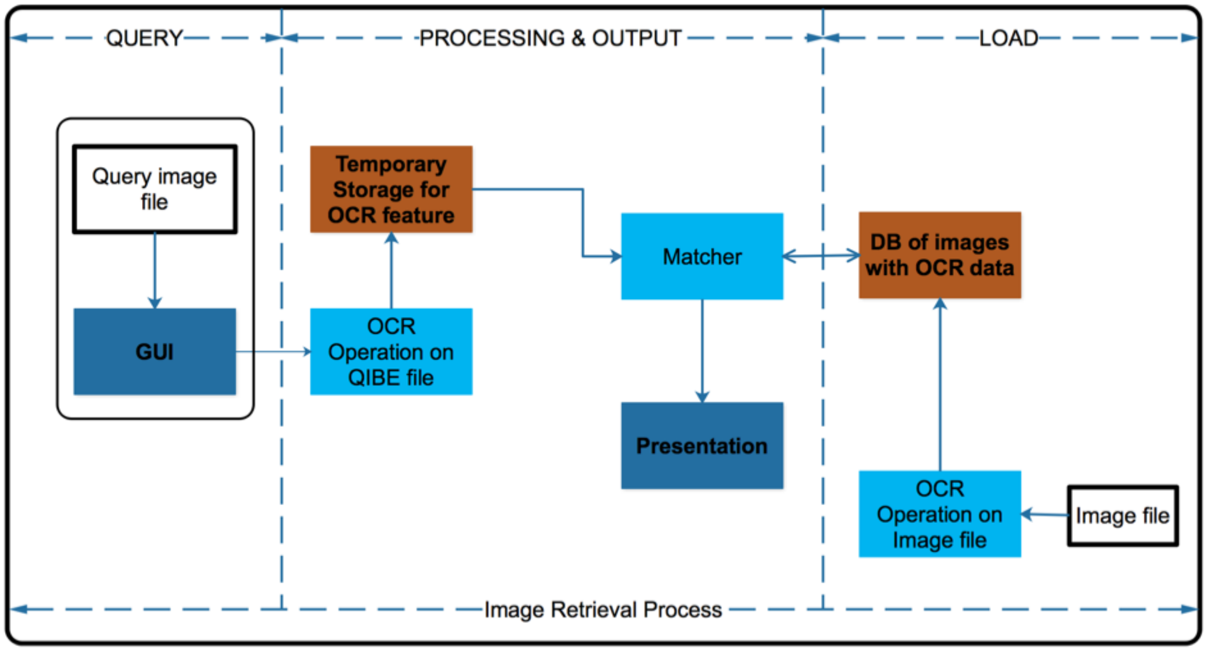

QUERY
PROCESSING & OUTPUT
LOAD
Query image file
GUI
Temporary Storage for OCR feature
OCR Operation on QIBE file
Matcher
Presentation
DB of images with OCR data
OCR Operation on Image file
Image file
Image Retrieval Process

Fig. 1. Conceptual Framework for the CBIR System.

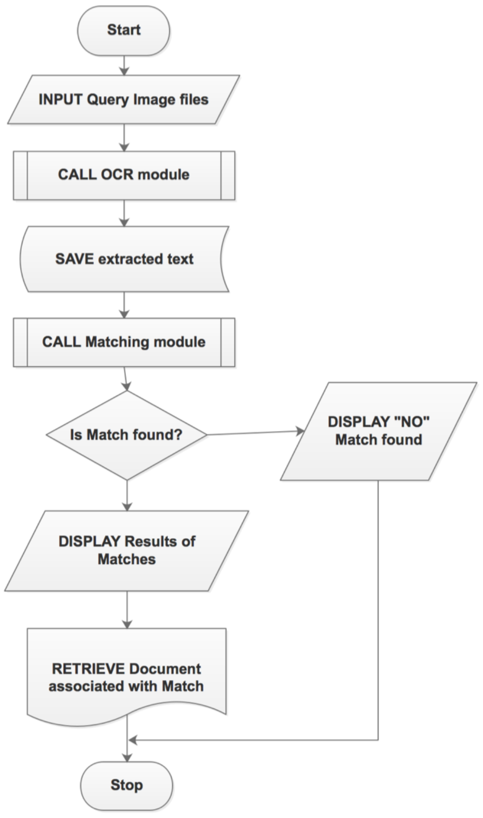


Fig. 2. Flowchart - Admin Side.

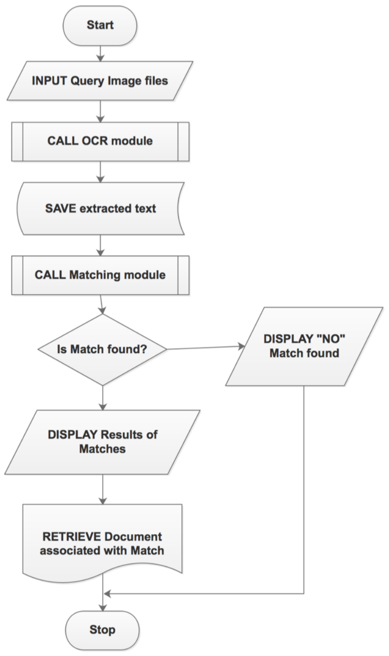


Fig. 3. Flowchart - user Side.

given to the Searcher [25].

2) *Processing and Output:* The query image file submitted is processed, and desired output is presented to the Searcher/User, as shown in Fig. 4. OCR operation is performed on the query image file to obtain the text feature of the file. This information is stored in a temporary storage location. The text features are sent to the matcher comparison. The Processing and Output section is between the front-end and the back-end (see Fig. 4). The processing and output part

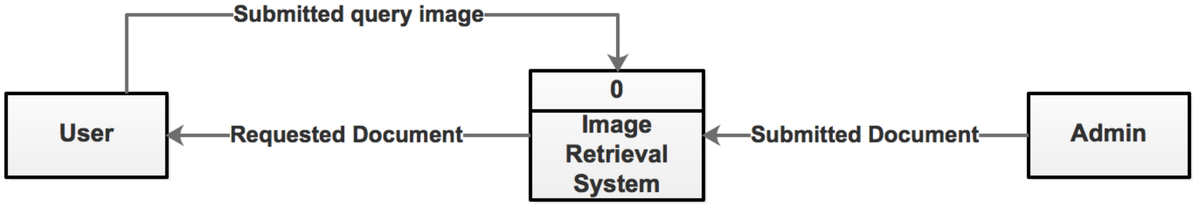


Fig. 4. Context Level Diagram of the CBIR System.

that acts as the *workhorse* for the system. The image upload by the user is processed in this section. Text matching to find the entire document and the display output is performed here. The Tesseract OCR engine evaluates the query image upload by the user, and the extracted text is saved in a temporary location. The content is saved in a string variable and compared to all extracted images uploaded by the Admin. The improved Levenshtein algorithm makes a comparison. The operation of the Levenshtein Algorithm on the implementation are discussed in Section IV. After the comparison is made the output is shown to the user. The outputs are ranked from the top to the bottom according to the similarity index found in the database, with the highest index at top. The user selects the appropriate link to download the document [26].

3) Load: The Load part stores information for Searcher/User retrieval. It serves as the database repository during the search process, which is similar to searching over the web. The image files are loaded into the system, then OCR operation is performed on the image to make the text recognisable. The recognised texts are stored with the image in the database. The Load part serves as the back-end to the system. It is in this section that the Administrator uploads the image files of a document to the system. Once the files have been uploaded the Tesseract OCR engine acts on it, and the extracted text is saved in the database. The file remains on the server but not saved in the database. However, the file path leading to where the file is saved is instead saved in the database. This makes the database lighter, hence, improving the system's performance.

IV. Analysis of the Tesseract OCR and Levenshtein Algorithm

A. Tesseract OCR

The primary purpose of the Tesseract OCR is to recognise characters in a given image. After the characters have been recognised, it is then extracted and saved. The characters extracted are group into words. A word is series of character without white space. Words recognised do have a confidence level. The confidence level of a word shows the accuracy level of the recognised word. Fig. 5 shows a sample image document that was uploaded to test the working of the OCR. Fig. 5 contains 595 words. At different resolutions, the number of words that the OCR recognised is as follows:

- Results of words at 2167 x 3064 pixel - 609 words
- Results of words at 724 x 1024 pixel - 91 words
- Results of words at 566 x 800 pixel - 20 words

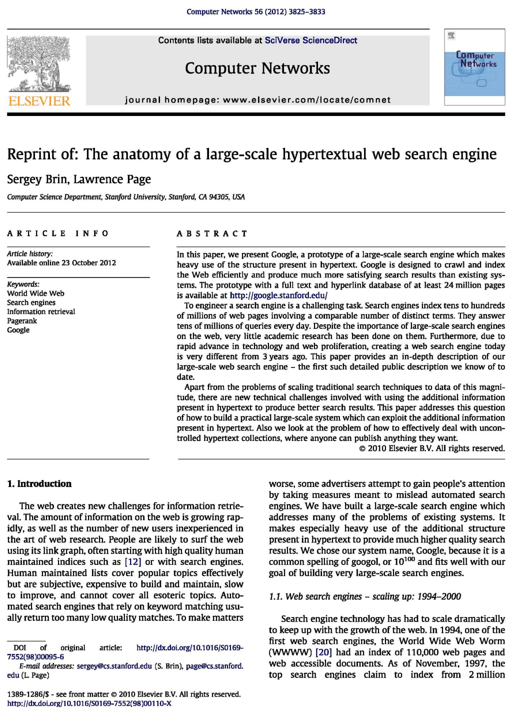
The 609 words recognised in the 2167 x 3064 is because the OCR recognise some punctuations as words. It is clear that the lower the resolution, the fewer words were recognised, and this is in line with the work done by [25]. Each word recognised by the OCR has a confidence level. The confidence of a word is the certainty of the accuracy of that word. Lower values indicate high certainty whereas, higher values indicate low certainty. Also, lower resolutions provided low confidence values even with the same words recognised in the different resolution. With the data obtained from the OCR, it shows that high resolution also provides better *meaningful words*. Meaningful words are all words that when checked in the document exists as it has been recognised. The observation made from the data can express as follows:

- $R \propto W_n$
- $W_n \propto W_m$
- Hence, $R \propto W_m$
- $W_n \propto C$
- Hence, $R \propto C$
  where R is the Resolution, W_n is the number of words recognised, W_m is the number of meaningful words, and C is the confidence of a word.

B. Levenshtein Algorithm

The Levenshtein algorithm, also known as the Edit-Distance algorithm was developed by Vladimir Levenshtein [27]. The algorithm provides information about the least number of edits needed to modify one string to obtain another string. Levenshtein algorithm measures similarity and matches approximate strings with fuzzy logic. Thus, the difference between the two strings is not presented as either absolutely true or false [28]. The distance between two character strings is the difference in the strings due to the number of deletions, insertions, or substitution. A distance of zero (0) indicates that the two strings are identical anything else means they are not the same. The more prominent the distance, the more unique the strings are [25]. In order to test how the system is working concerning the algorithm, four (4) different files (image documents) which sum up to 13 pages was uploaded from the administrator's side of the system.

The first file constituted the first four pages in the database; the second file constituted the next three pages, the third file the next two pages and the fourth file the last four pages. From the user's side, a query image was submitted to the system. The query image that was used in the first instance is shown in Fig. 5. The second and third instance made use of portions of Fig. 5, these portions are shown in Fig. 6 and Fig. 7, respectively.



Computer Networks 56 (2012) 3825–3833

Contents lists available at SciVerse ScienceDirect

Computer Networks

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

ELSEVIER

Reprint of: The anatomy of a large-scale hypertextual web search engine

Sergey Brin, Lawrence Page

Computer Science Department, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Available online 23 October 2012

Keywords:
World Wide Web
Search engines
Information retrieval
Pagerank
Google

ABSTRACT

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from 3 years ago. This paper provides an in-depth description of our large-scale web search engine – the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections, where anyone can publish anything they want.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as [12] or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search engines.

1.1. Web search engines – scaling up: 1994–2000

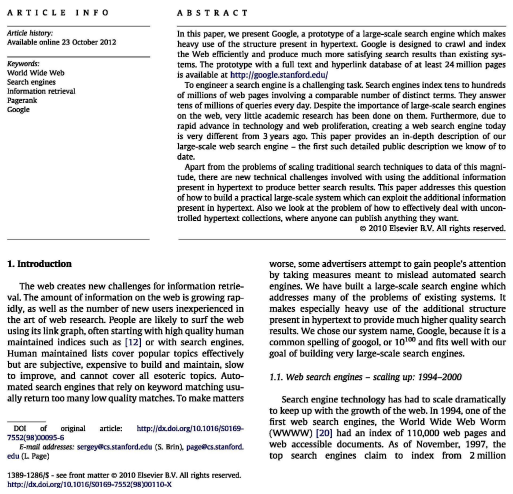
Search engine technology has had to scale dramatically to keep up with the growth of the web. In 1994, one of the first web search engines, the World Wide Web Worm (WWWW) [20] had an index of 110,000 web pages and web accessible documents. As of November, 1997, the top search engines claim to index from 2 million

DOI of original article: http://dx.doi.org/10.1016/S0169-7552(98)00095-6

E-mail addresses: sergey@cs.stanford.edu (S. Brin), page@cs.stanford.edu (L. Page)

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Fig. 5. Sample Image Document.



ARTICLE INFO

Article history:
Available online 23 October 2012

Keywords:
World Wide Web
Search engines
Information retrieval
Pagerank
Google

ABSTRACT

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from 3 years ago. This paper provides an in-depth description of our large-scale web search engine – the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections, where anyone can publish anything they want.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as [12] or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search engines.

1.1. Web search engines – scaling up: 1994–2000

Search engine technology has had to scale dramatically to keep up with the growth of the web. In 1994, one of the first web search engines, the World Wide Web Worm (WWWW) [20] had an index of 110,000 web pages and web accessible documents. As of November, 1997, the top search engines claim to index from 2 million

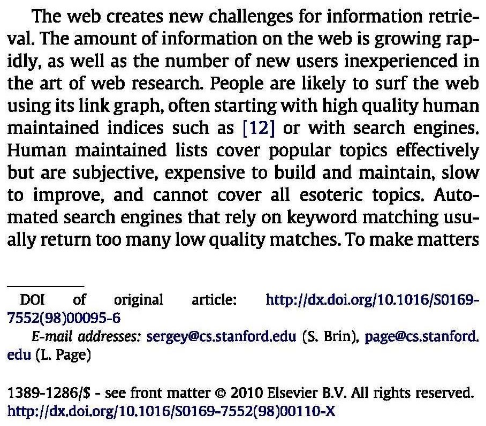
DOI of original article: http://dx.doi.org/10.1016/S0169-7552(98)00095-6

E-mail addresses: sergey@cs.stanford.edu (S. Brin), page@cs.stanford.edu (L. Page)

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Fig. 6. Partial Image of Fig. 5 for Query.

In Fig. 8, we show the comparison ratio that was attained based on the three different query images that were used. The query image Fig. 6 that was used in the first query is the entire page 1 of the first file in the database. This return, the highest comparison ratio of 1, in other words, 100% match, as depicted in Fig. 9. The second and the third query as already indicated, was based on partial images of Fig. 6, and the highest comparison ratio return were 0.88 and 0.22, respectively. From these results, it is clear that the more words found in a query image, the higher the comparison ratio. Further tests indicate that when a file of a particular image document is used as the search query, the results that are the comparison ratio were close to 1. In Fig. 9, the individual pages of *document1* were uploaded as the search query, the results



The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as [12] or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters

DOI of original article: http://dx.doi.org/10.1016/S0169-7552(98)00095-6

E-mail addresses: sergey@cs.stanford.edu (S. Brin), page@cs.stanford.edu (L. Page)

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Fig. 7. Partial Image of Fig. 6 for Query.

indicate that page 1 returned a comparison ratio of 0.99, page 2 returned 0.92, page 3 returned 0.98 and page 4 returned 0.99. Similar results were obtained from the experiment conducted, and these are shown in Fig. 10, 11, and 12.

Files that cannot get a comparison ratio of 0.15 and above in any of its pages is excluded from the search results. So in the third query were only the first file had a page with a comparison more significant than the 0.15 criteria, only the first file was return as the search results. However, the first and second query returned all the files as the search result since they were the page (s) in each files having a comparison ratio 0.15. Other tests were done with different query image, and these test yielded similar trends as discussed above. Also, future work can be that of having a crawler attached to the system that will crawl the web if its implementation is to be used universally on the internet. This crawler's purpose is to get images with their URL. The system can then extract the text content of these images and then link them to the URL where they were gotten from so that when a user submits a query image, the search results will be the URL links.

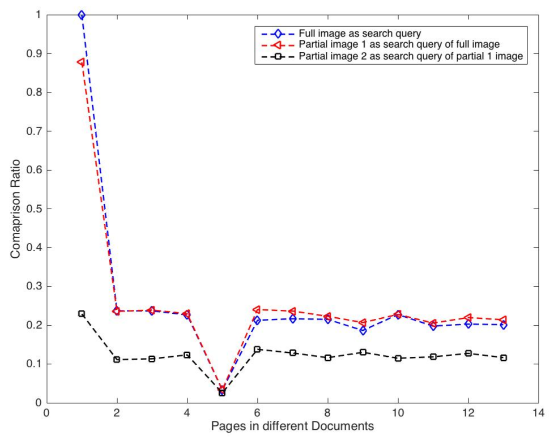


Fig. 8. Comparison Ratio of Fig. 5, 6, and 7.

In Fig. 13, the full page of an image *sample image - file 4* were uploaded as the search query, the results indicated in

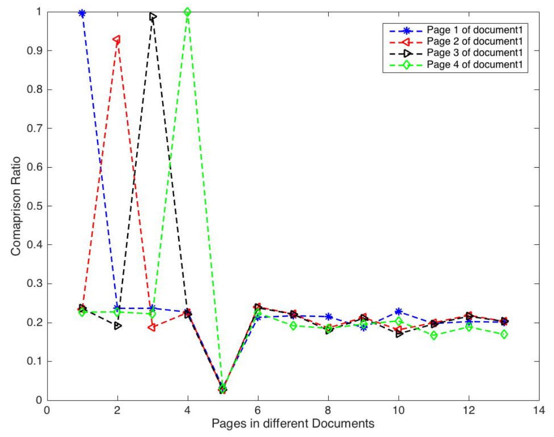


Fig. 9. Comparison Ratio of Pages 1 to 3 of Different Document.

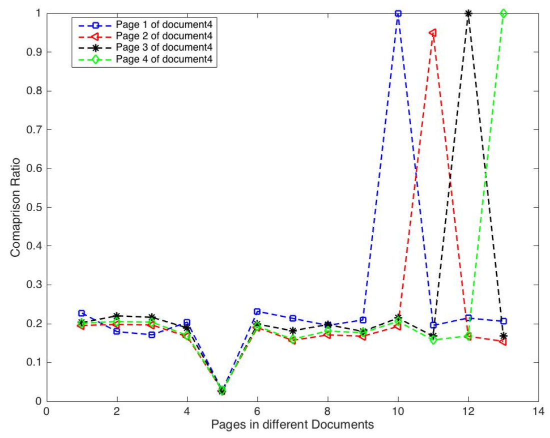


Fig. 12. Comparison Ratio of Pages 1 to 4 of Different Document.

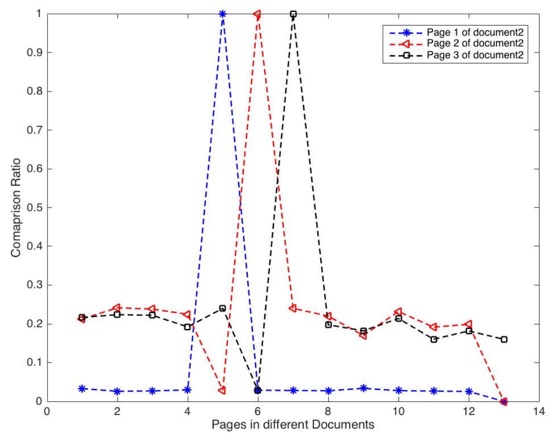
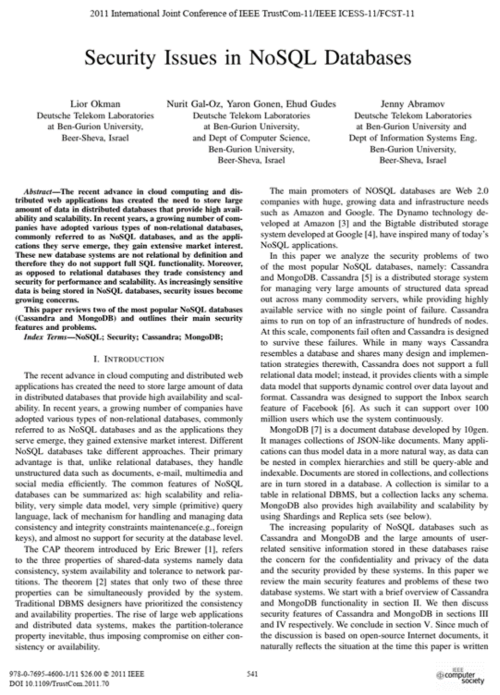


Fig. 10. Comparison Ratio of Pages 1 to 3 in the Database.



2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11

Security Issues in NoSQL Databases

Lior Okman
Deutsche Telekom Laboratories
at Ben-Gurion University,
Beer-Sheva, Israel

Nurit Gal-Oz, Yaron Gonen, Ehud Gudes
Deutsche Telekom Laboratories
at Ben-Gurion University,
and Dept of Computer Science,
Ben-Gurion University,
Beer-Sheva, Israel

Jenny Abramov
Deutsche Telekom Laboratories
at Ben-Gurion University and
Dept of Information Systems Eng.
Ben-Gurion University,
Beer-Sheva, Israel

***Abstract*—The recent advance in cloud computing and distributed web applications has created the need to store large amount of data in distributed databases that provide high availability and scalability. In recent years, a growing number of companies have adopted various types of non-relational databases, commonly referred to as NoSQL databases, and as the applications they serve emerge, they gain extensive market interest. These new database systems are not relational by definition and therefore they do not support full SQL functionality. Moreover, as opposed to relational databases they trade consistency and security for performance and scalability. As increasingly sensitive data is being stored in NoSQL databases, security issues become growing concerns.**

This paper reviews two of the most popular NoSQL databases (Cassandra and MongoDB) and outlines their main security features and problems.

***Index Terms*—NoSQL; Security; Cassandra; MongoDB;**

I. Introduction

The recent advance in cloud computing and distributed web applications has created the need to store large amount of data in distributed databases that provide high availability and scalability. In recent years, a growing number of companies have adopted various types of non-relational databases, commonly referred to as NoSQL databases and as the applications they serve emerge, they gained extensive market interest. Different NoSQL databases take different approaches. Their primary advantage is that, unlike relational databases, they handle unstructured data such as documents, e-mail, multimedia and social media efficiently. The common features of NoSQL databases can be summarized as: high scalability and reliability, very simple data model, very simple (primitive) query language, lack of mechanism for handling and managing data consistency and integrity constraints maintenance(e.g., foreign keys), and almost no support for security at the database level.

The CAP theorem introduced by Eric Brewer [1], refers to the three properties of shared-data systems namely data consistency, system availability and tolerance to network partitions. The theorem [2] states that only two of these three properties can be simultaneously provided by the system. Traditional DBMS designers have prioritized the consistency and availability properties. The rise of large web applications and distributed data systems, makes the partition-tolerance property inevitable, thus imposing compromise on either consistency or availability.

The main promoters of NOSQL databases are Web 2.0 companies with huge, growing data and infrastructure needs such as Amazon and Google. The Dynamo technology developed at Amazon [3] and the Bigtable distributed storage system developed at Google [4], have inspired many of today's NoSQL applications.

In this paper we analyze the security problems of two of the most popular NoSQL databases, namely: Cassandra and MongoDB. Cassandra [5] is a distributed storage system for managing very large amounts of structured data spread out across many commodity servers, while providing highly available service with no single point of failure. Cassandra aims to run on top of an infrastructure of hundreds of nodes. At this scale, components fail often and Cassandra is designed to survive these failures. While in many ways Cassandra resembles a database and shares many design and implementation strategies therewith, Cassandra does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format. Cassandra was designed to support the Inbox search feature of Facebook [6]. As such it can support over 100 million users which use the system continuously.

MongoDB [7] is a document database developed by 10gen. It manages collections of JSON-like documents. Many applications can thus model data in a more natural way, as data can be nested in complex hierarchies and still be query-able and indexable. Documents are stored in collections, and collections are in turn stored in a database. A collection is similar to a table in relational DBMS, but a collection lacks any schema. MongoDB also provides high availability and scalability by using Shardings and Replica sets (see below).

The increasing popularity of NoSQL databases such as Cassandra and MongoDB and the large amounts of user-related sensitive information stored in these databases raise the concern for the confidentiality and privacy of the data and the security provided by these systems. In this paper we review the main security features and problems of these two database systems. We start with a brief overview of Cassandra and MongoDB functionality in section II. We then discuss security features of Cassandra and MongoDB in sections III and IV respectively. We conclude in section V. Since much of the discussion is based on open-source Internet documents, it naturally reflects the situation at the time this paper is written

978-0-7695-4600-1/11 $26.00 © 2011 IEEE
DOI 10.1109/TrustCom.2011.70

541

IEEE computer society

Fig. 13. Sample Image - File 4.

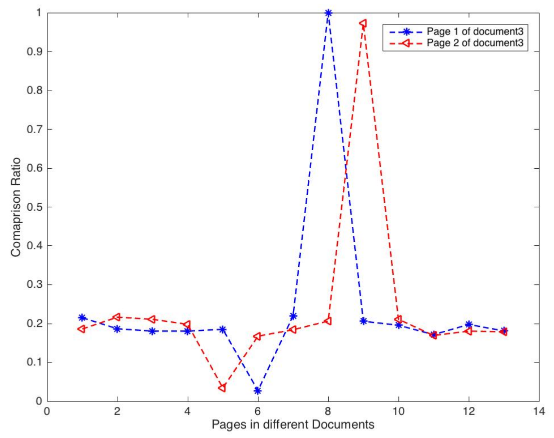


Fig. 11. Comparison Ratio of Page 1 and 2 of Different Document.

Table II shows the various comparison ratios.

V. Conclusion

The paper employed the use of the Tesseract OCR engine and an improved Levenshtein Algorithm in content-based image retrieval system to extract text from an image and use the text-image to search for the full document containing the text. The text content of the image was used as the base. The system as it stands now can be used for the storing and retrieving of image documents. Just by providing a query image, the right document will be provided to the User once it exists in the database. This system can be employed in areas where

TABLE II. SEARCH RESULTS USING FULL IMAGE FILE (PAGE 1 OF FILE 4 OF FIG. 13).

Image File Pages	Comparison Ratio
File 1 Pg 1	0.227470472
File 1 Pg 2	0.179640725
File 1 Pg 3	0.171515763
File 1 Pg 4	0.2040605
File 2 Pg 1	0.027760513
File 2 Pg 2	0.231821
File 2 Pg 3	0.21379739
File 3 Pg 1	0.195773765
File 3 Pg 2	0.2104827
File 4 Pg 1	1
File 4 Pg 2	0.196395278
File 4 Pg 3	0.214626059
File 4 Pg 4	0.204474822

there is a need to digitised documents into images for onward retrieval base on the query image. From the experimental results, our approach a showed 100% match when the query image matches its page in the database; otherwise, lower values would be obtained when the extracted text does relate to other text.

VI. FUTURE WORK

In this section, we discuss some important research directions that require additional attention to improve content-based image retrieval systems.

A. Machine Learning Algorithms and Artificial Neural Networks

In most CBIR systems, the query techniques employed by authors are query by example to obtain the relevant features. These features are not adequate in themselves. Hence, advanced knowledge-based techniques and approaches capable of extracting multiple features from documents may efficiently improve the search criteria. These machine learning algorithms and artificial neural networks methods may also validate the extracted features, perform classification on the images and select relevant aspects for the search query. By this means, query features that are not capable of providing accurate results will be eliminated.

B. Multi-level Text Matching

Current approaches in CBIR systems based on text features rely mainly on single-level text matching. It is recommended that future CBIR systems implement a multi-level text matching capable of improving retrieval efficiency to improve the efficiency of the document retrieved.

C. Searching by Cluster of Words (or paragraph)

One of the significant limitations for searching with the current approach is lower image resolution. Hence, instead of word by word matching, we recommend that future work focuses on a cluster of words (or paragraph) matching, which will significantly increase the accuracy.

D. Gathering Data using Crawlers

A crawler can be attached to the system that will crawl the web if the system is implemented and used universally on the web. This crawler's purpose is to get images with their respective URL. The system can then extract the text content of these images and then link them to the URL where they were gotten from so that when a user submits a query image, the search results will be the URL links.

ACKNOWLEDGMENT

The authors would like to thank colleagues who supported us in diverse ways to get this manuscript ready for publication.

REFERENCES

[1] H. M. Al-Barhamtoshy, K. M. Jambi, S. M. Abdou, and M. A. Rashwan, "Arabic documents information retrieval for printed, handwritten, and calligraphy image," *IEEE Access*, vol. 9, pp. 51 242–51 257, 2021.

[2] K. Pustu-Iren, G. Bruns, and R. Ewerth, "A multimodal approach for semantic patent image retrieval," 2021.

[3] P. Ghadekar, S. Kaneri, A. Undre, and A. Jagtap, "Digital image retrieval based on selective conceptual based features for important documents," in *Evolutionary Computing and Mobile Sustainable Networks*. Springer, 2021, pp. 569–579.

[4] K. M. Lakshmi *et al.*, "An efficient telugu word image retrieval system using deep cluster," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 11, pp. 3247–3255, 2021.

[5] L. Yang, M. Gong, and V. K. Asari, "Diagram image retrieval and analysis: Challenges and opportunities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 180–181.

[6] W. Zhou, H. Li, and Q. Tian, "Recent advance in content-based image retrieval: A literature survey," *arXiv preprint arXiv:1706.06064*, 2017.

[7] A. Jain, "The role and importance of search engine and search engine optimization," *International Journal of emerging trends & technology in computer science*, vol. 2, no. 3, pp. 99–102, 2013.

[8] N. Höchstötter and D. Lewandowski, "What users see–structures in search engine results pages," *Information Sciences*, vol. 179, no. 12, pp. 1796–1812, 2009.

[9] A. V. Singh, "Content-based image retrieval using deep learning," 2015.

[10] B. Wang, X. Zhang, and N. Li, "Relevance feedback technique for content-based image retrieval using neural network learning," in *2006 International Conference on Machine Learning and Cybernetics*. IEEE, 2006, pp. 3692–3696.

[11] A. Rashno and E. Rashno, "Content-based image retrieval system with most relevant features among wavelet and color features," *arXiv preprint arXiv:1902.02059*, 2019.

[12] R. R. Saritha, V. Paul, and P. G. Kumar, "Content based image retrieval using deep learning process," *Cluster Computing*, vol. 22, no. 2, pp. 4187–4200, 2019.

[13] K. Zagoris, S. A. Chatzichristofis, N. Papamarkos, and Y. S. Boutalis, "img (anaktisi): A web content based image retrieval system," in *2009 Second International Workshop on Similarity Search and Applications*. IEEE, 2009, pp. 154–155.

[14] K.-M. Wong, K.-W. Cheung, and L.-M. Po, "Mirror: an interactive content based image retrieval system," in *2005 IEEE International Symposium on Circuits and Systems*. IEEE, 2005, pp. 1541–1544.

[15] F. Karamzadeh and M. A. Azgomi, "An automated system for search and retrieval of trademarks," in *Proceedings of the 11th International Conference on Electronic Commerce*, 2009, pp. 374–377.

[16] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. A. Reznik, "Mobile visual search: Architectures, technologies, and the emerging mpeg standard," *IEEE MultiMedia*, vol. 18, no. 3, pp. 86–94, 2011.

[17] G. A. Montazer and D. Giveki, "Content based image retrieval system using clustered scale invariant feature transforms," *Optik*, vol. 126, no. 18, pp. 1695–1699, 2015.

[18] S. A. Bakar, M. S. Hitam, and W. N. J. H. W. Yussof, "Content-based image retrieval using sift for binary and greyscale images," in *2013 IEEE International Conference on Signal and Image Processing Applications*. IEEE, 2013, pp. 83–88.

[19] P. Liu, J.-M. Guo, C.-Y. Wu, and D. Cai, "Fusion of deep learning and compressed domain features for content-based image retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5706–5717, 2017.

[20] R. S. Choras, "Image feature extraction techniques and their applications for cbir and biometrics systems," *International journal of biology and biomedical engineering*, vol. 1, no. 1, pp. 6–16, 2007.

[21] M. Koistinen, K. Kettunen, and J. Kervinen, "How to improve optical character recognition of historical finnish newspapers using open source tesseract ocr engine," *Proc. of LTC*, pp. 279–283, 2017.

[22] H. Xu, J.-y. Wang, and L. Mao, "Relevance feedback for content-based image retrieval using deep learning," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 2017, pp. 629–633.

[23] I. Sommerville, "Software engineering 9th edition," *ISBN-10*, vol. 137035152, p. 18, 2011.

[24] M. Smitha, P. Antony, and D. Sachin, "Document image analysis using imagemagick and tesseract-ocr," *International Advanced Research Journal in Science, Engineering and Technology (IARJSET)*, vol. 3, pp. 108–112, 2016.

[25] S. Smith, "Overview of asp .net core mvc," *Microsoft. Last modified January*, vol. 7, 2018.

[26] J. S. Valacich, J. F. George, and J. S. Valacich, *Modern systems analysis and design*. Pearson Education Limited London, 2017.

[27] G. Naganjaneyulu, A. Narasimhadhan, and K. Venkatesh, "Performance evaluation of ocr on poor resolution text document images using different pre processing steps," in *TENCON 2014-2014 IEEE Region 10 Conference*. IEEE, 2014, pp. 1–4.

[28] M. Gilleland *et al.*, "Levenshtein distance, in three flavors," *Merriam Park Software: http://www. merriampark. com/ld. htm*, 2009.