IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 27, 2019, accepted July 15, 2019, date of publication July 22, 2019, date of current version August 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930339

DoCA: A Content-Based Automatic Classification System Over Digital Documents

SÜLEYMAN EKEN, HOUSSEM MENHOUR, AND KÜBRA KÖKSAL
Department of Computer Engineering, Umuttepe Campus, Kocaeli University, 41380 Kocaeli, Turkey
Corresponding author: Süleyman Eken (suleyman.eken@kocaeli.edu.tr)

ABSTRACT Regardless of industry, the overload of information facing most organizations today is a drain on both individuals and the enterprise itself. The increasing volume of this information, which is stored in different electronic formats, requires new sophisticated systems to analyse and classify them. In this paper, we attempt to implement a framework Document Classification and Analysis (DoCA) that can simplify and automate such tasks for different file types, namely: office documents (text, spreadsheets, and presentations), scanned documents (images and PDFs), multimedia files (video and audio). Each file type requires different methods for pre-processing, analysis, and classification. The efficiency and feasibility of the DoCA are examined on HAVELSAN dataset and accuracy of different tasks shows that the DoCA is a promising tool for document analysis and classification.

INDEX TERMS Document analysis, document classification, OCR, video-audio analysis.

I. INTRODUCTION

Due to the increasing amount of electronic documents and the rapid growth of the World Wide Web, automatic document analysis and classification became very important for information organization and storage. Document analysis is a form of quantitative research to systematically evaluate documents and find meaning in them [1]. Document classification is an age-old problem in information science with key importance in a variety of applications. In general, it is the task of assigning one or more labels or categories to a given document. This task can be tackled in different ways depending on the type of the document.

For text documents, pre-processing is a must to get reliable results, that includes tokenization, stemming and vectorisation. Document classification is an example of Machine Learning (ML) in the form of Natural Language Processing (NLP). The classification can be based either on supervised or unsupervised learning. In the first case, categories are predefined and assigned to the appropriate documents in the training dataset, which is used to train a model that in turn can predict the categories of new documents. In the second case, documents must be clustered automatically without the need for predefined classes.

The associate editor coordinating the review of this manuscript and approving it for publication was Sabah Mohammed.

For images and PDFs that contain text, Optical Character Recognition (OCR) can be used to extract text from them, which allows us to treat them as text documents for classification or keyword search tasks. In case of image-based search, however, OCR can be skipped and template matching algorithms can be leveraged for the task. Template matching is a method for locating a template image in a larger image. SIFT is an example of algorithms that allows performing such tasks. The results of template matching and those of text search (after OCR), served as the basis for classification in this file type category.

For audio files, classification can be in the form of gender identification or speech/music segmentation for example. A common technique to achieve that is converting a raw waveform into a spectrogram that can be then used to feed a Convolutional Neural Network (CNN) [2], [3]. For video files, the problem boils down to processing a sequence of frames, then different neural networks architectures can be used to implement the classification [4]. In this work a simpler approach is taken; we needed to determine whether a video is from security camera footage or not.

The motivation for this study was a competition [5] organized by HAVELSAN [6], and PARDUS [7] which we won first place in. The biggest challenge we faced in this project is having to work with a small and unlabeled dataset, while it mimics real life environments, it limits our options on how to tackle the problem. In this study, we create a framework which provides different institutions and organizations with a tool to analyze different file types and classify them. To the best of our knowledge, there is no encompassing study about the document analysis and classification of all these file types in a hierarchical way: 1) In the first level, we deal with document diff and similarity analysis for text-based file types, template matching for image files, and content analysis for audio and video files. 2) In the second level, the construction of the tree structure of the relevant files, the similarity of each other and the determination of the change in the parent-child relationship are carried out, a new version of any file will be saved in the database as a child of the original file. In other words, we deal with file and folder watchdog analysis. 3) In the last level, automatic document classification is implemented; after classifying already existing documents into their respective classes, a background task will detect any newly created document and automatically classify it according to the previously established classes. The different algorithms used for these tasks were well-established ones, so our work focused on picking the best and leveraging them together as one coherent system. Section III will dig deeper into the technical details of each of these tasks and our choice of algorithms which, combined together, create an effective tool for big organizations to help them organize and deal with their digital clutter, as well as tracking any changes in important documents.

The rest of this paper is organized as follows: Section II presents a review of the different tasks in the framework. Section III describes the presented system architecture with its submodules. Section IV deals with the realized experiments. In the last section, we discuss the research results and we propose some future research directions.

II. RELATED WORKS

In this section, a summary of related works of different tasks of DoCA will be presented.

A. DOCUMENT DIFF AND SIMILARITY ANALYSIS

Google Diff-Match-Patch [8] implements Myer's diff algorithm which is generally considered to be the best general-purpose diff. It was first presented in 1986 [9] and has a variety of applications, including UNIX Diff utility, and Google's Diff-Match-Patch library. Myer's algorithm builds on earlier notable work on Longest Common Subsequences (LCS) [10], [11].

String matching is the process of finding strings that match a given pattern approximately (rather than exactly), like literally. Hence it is also known as approximate string matching. Usually, the pattern that these strings are matched against is another string. For string matching, Levenshtein distance is used [12]. Since then, considerable researches have been done on the subject of the string-to-string correction problem [13]–[15].

Other researchers delve into more specific problems such as the case with semantic interoperability between IoT devices and users [16].

B. TEMPLATE MATCHING

Over the years, a wide variety of features detectors and image matching frameworks have been proposed. Building on previous works like Harris' on edge detection [17], Lowe came up with Scale Invariant Feature Transform (SIFT) [18], [19]. Later work by Muja and Lowe introduced FLANN [20]–[22], a library for performing fast approximate nearest neighbor searches in high dimensional spaces, which can be used in one of SIFT steps for better performance. Ling and Jacobs proposed a deformation invariant image matching [23]. Bay *et al.* proposed Speeded-Up Robust Features (SURF) [24] and Simo-Serra, Torras and Moreno-Noguer proposed Deformation and Light Invariant (DaLI) descriptor [25], [26]. There is also a significant number of works focusing on studying and evaluating the performance of some descriptors [27]–[30].

C. AUDIO CONTENT ANALYSIS

Audio data can be useful for analysis. Indexing music collections according to their audio features, recommending music for radio channels, similarity search for audio files, speech processing and synthesis are a few examples of the potential applications of audio processing [31]. Audio classification is also a fundamental problem in the field of audio processing. The task is essential to extract features from the audio, and then identify which class it belongs to. Many useful applications pertaining to audio classification can be found in the wild-such as genre classification, instrument recognition and artist identification [32], speech recognition [33] and gender classification [34].

D. VIDEO CONTENT ANALYSIS

The recent trend is to rely on Neural Networks (NN) for video classification by extracting what is often called as feature descriptors from frames. Researchers have accomplished this with several methods. Joe *et al.* [35] discussed two methods: The first uses CNNs to extract frame-based features through time, and the second treats the video as an ordered sequence of frames fed to a recurrent neural network that uses Long Short-Term Memory (LSTM) cells which are connected to the output of the underlying CNN. However, training with LSTM is difficult and expensive. Graham *et al.* improved on that by introducing Convolutional Drift Networks (CDNs) for video classification [36], their method relies on combining CNNs and ESNs. A trainable end-to-end architecture that requires no hand-crafted features.

The area of object recognition in surveillance videos also has seen a considerable amount of research. Nascimento and Marques [37] compared five different algorithms in this field. Their focus was moving object detection, but these algorithms can be leveraged for non-moving objects as well. More specifically, research on detecting abandoned luggage has shown results as well, either by means of background subtraction [38] or by using CNNs for better results [39].

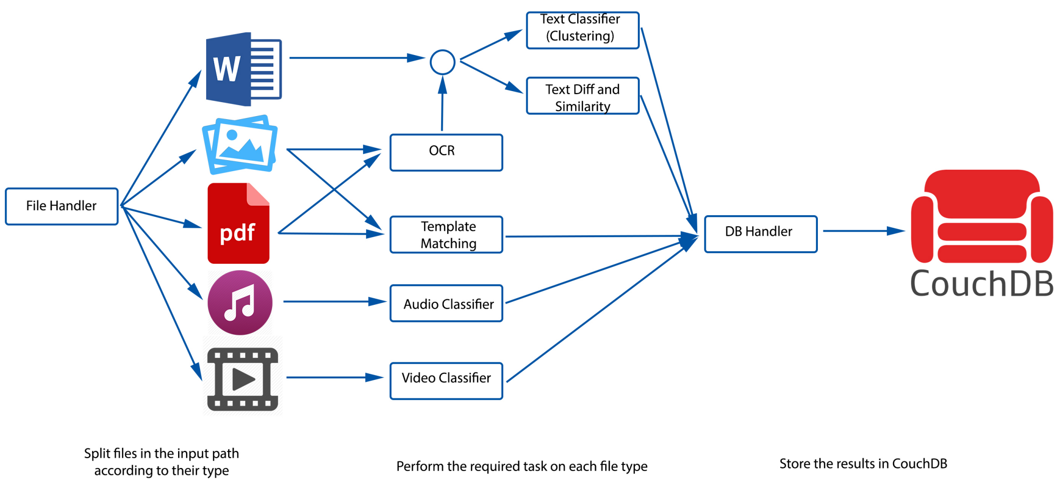

Text Classifier (Clustering)
Text Diff and Similarity
OCR
File Handler
Template Matching
DB Handler
Audio Classifier
Video Classifier
CouchDB
Split files in the input path according to their type
Perform the required task on each file type
Store the results in CouchDB

FIGURE 1. **Overview of the DoCA architecture.**

E. FILE AND FOLDER WATCHDOG ANALYSIS

There are several system event monitors and loggers that can allow us to identify and track any changes to files and folders. Among them, inotify-tools [40] is the most popular one which is a C library and a set of command-line programs for Linux providing a simple interface to inotify, the Linux kernel subsystem that acts to allow noticing changes to the file system. There is also Listen gem [41] and Guard [42], offering similar functionality with extending it to MacOS and Windows and implemented using Ruby. For Python users, there is pyinotify [43] on Linux and the cross-platform PyFilesystem [44], an abstraction layer for Python's file system. The most recent and most popular choice, however, is Watchdog [45].

F. DOCUMENT CLASSIFICATION

When it comes to text data classification, we can split it into two different categories; classification based on a model trained on labeled data first, or clustering without the need for any pre-labelled dataset. Both cases have seen significant research. That includes research on word representation in vector space [46], on labeled text classification using various techniques ranging from the simplest Naive Bayes (NB) and TF-IDF (Term Frequency-Inverse Document Frequency) classifiers [47] to NNs passing by Support Vector Machines (SVM) based ones [48], [49]. There are also works on the use of genetic algorithms for the task [50]. When it comes to clustering, in 1996 Martin Ester *et al.* proposed density-based spatial clustering of applications with noise (DBSCAN) [51], it finds core samples of high density and expands clusters from them. DBSCAN is good for data which contains clusters of similar density. Frey introduced a newer clustering algorithm called Affinity Propagation (AP) [52], [53]. More recent work also explores techniques like Sampling-PSO-K-means [54].

III. SYSTEM ARCHITECTURE

In this section, the system architecture is presented, with each subsection including details of algorithms for different tasks of DoCA. The proposed system is implemented using Python programming language, this choice is mainly due to the ease of use and availability of 3rd party libraries to reduce the amount of work needed. A high-level representation of the DoCA is shown in Figure 1.

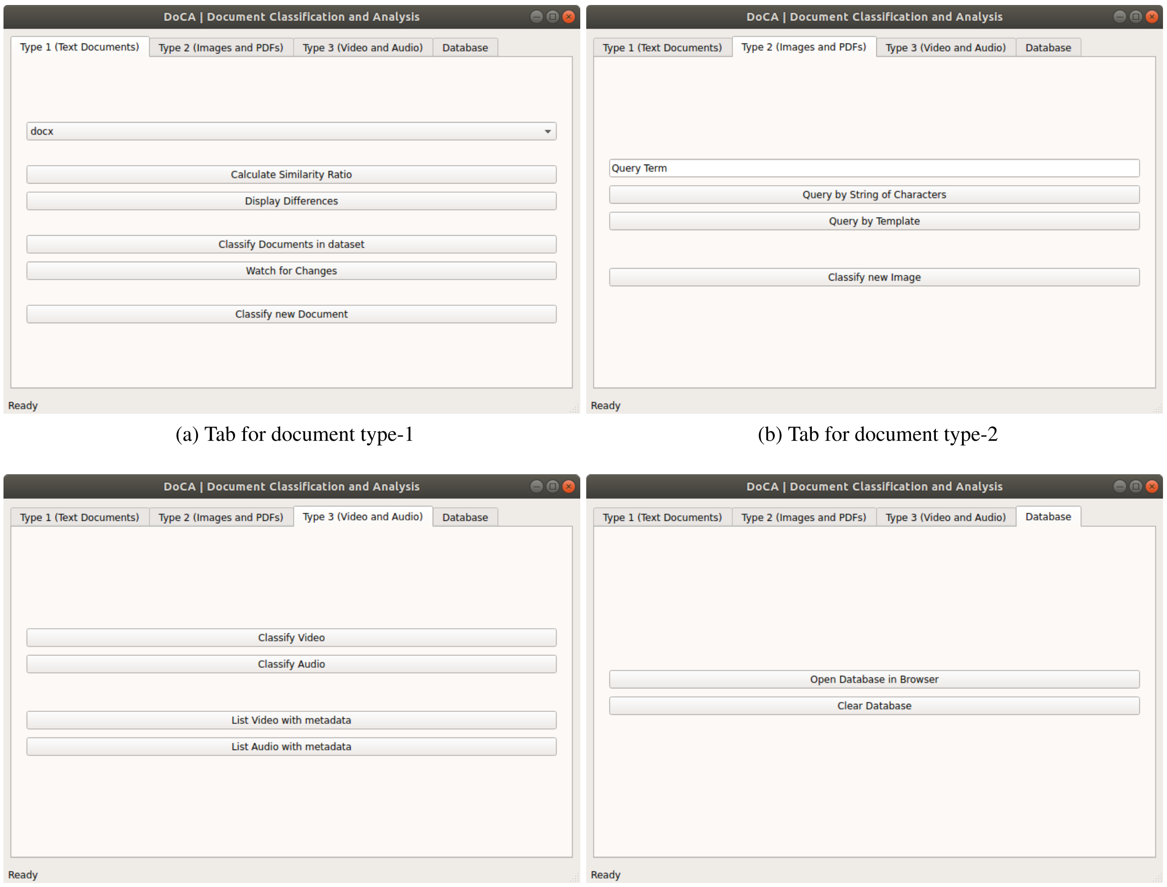
In addition to what the Figure 1 shows, there is a watchdog functionality that can run in the background and automatically perform the appropriate tasks on any new files. There is also a search functionality on image text content offered through the GUI as shown in Figure 2.

A. COUCHDB INTEGRATION

NoSQL, which stands for ''not only SQL'' is an alternative to traditional tabular relations used in relational databases in which data schema is carefully designed before the database is built making any later changes very difficult. NoSQL databases are especially useful and increasingly chosen for working with large sets of distributed data and real-time web application [55]. Among the key features of NoSQL approach, we can point out: simplicity of design, horizontal scaling to clusters of machines, superior performance, more flexibility [56].

We opted to adopt Apache CouchDB as our NoSQL database of DoCA (as seen in Figure 3). CouchDB has document-oriented NoSQL database architecture. Which means it differs from relational databases in the sense that RDBs store data in separate tables where a single object may be spread across several tables, while document databases dedicate a single instance, or document, for each object where it stores all its information.

CouchDB is implemented in Erlang, handles concurrency with a form of multi-version concurrency control (MVCC).



(a) Tab for document type-1

(b) Tab for document type-2

(c) Tab for document type-3

(d) Tab for document type-4

FIGURE 2. GUI design of DoCA. Tasks are grouped under tabs for each of the document file types.

It offers an intuitive RESTful HTTP/JSON API and uses MapReduce of JavaScript functions as the query method.

Other features include document-level ACID (Atomicity, Consistency, Isolation, Durability) semantics and multi-master replication, which allows it to scale across machines to build high-performance systems. A database handler is created to facilitate running any operation across the whole DoCA system.

B. DOCUMENT DIFF AND SIMILARITY ANALYSIS

Google Diff-Match-Patch [8] is used for document difference and similarity analysis. It implements Myer's diff algorithm which is generally considered to be the best general-purpose diff algorithm [9]. An example of its output is shown in Figure 4

Meyer's diff works by finding the LCS in an edit graph. In Figure 5, we see an edit graph for two sequences, A = abcabba and B = cbabac. The points (x, y) for which ax = by are called match points. A trace of length L is a sequence of L match points which is the number of diagonal edges in the resulting path.

For string matching; i.e. similarity analysis, Fuzzy-Wuzzy library [58] is used. It implements Levenshtein distance [12]. Fuzzy string matching is the process of finding strings that match a given pattern.

C. TEMPLATE MATCHING

For the task of template matching, the SIFT algorithm is used. Even though it is slower to run, it offers unparalleled accuracy. SIFT is an image descriptor for image-based matching and object recognition. The SIFT descriptor is invariant to translations, rotations and scaling transformations in the image domain and robust to moderate perspective transformations and illumination variations. There are five main steps involved in the SIFT algorithm, the following is a high-level summarization of these steps:

- Scale-space [59] Extrema Detection: Difference of Gaussian (DoG) is found for the image with various σ values. It is obtained by subtracting one blurred version of an original image from another, less blurred version. This process is done for different octaves of the image in the Gaussian Pyramid. An octave being the set of images generated by progressively blurring out an image. Once the DoGs are found, images are searched for local extrema over scale and space, which represent potential keypoints locations.

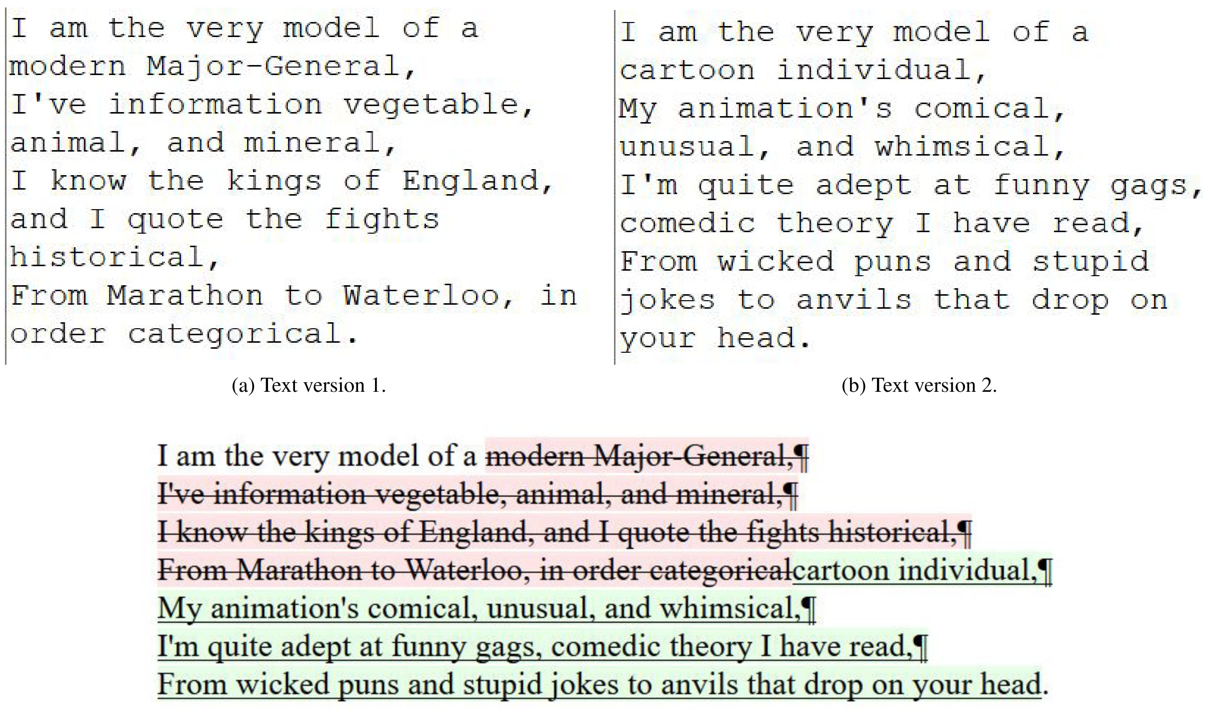
```
{"vid 20": {
  "id": "0020.mp4",
  "location": "/home/dataset/vid/0020.mp4",
  "class": "Security",
  "metadata": {
      "duration": "50s",
      "fps": "Open",
      "resolution": "1080*720",
  }
}}
{"docx 01": {
  "id": "0001.docx",
  "location": "/home/dataset/docx/0001.docx",
  "content": "<content>",
  "version history": {
                              0001.docx.01,
                              0001.docx.02},
  "metadata": {
      "created": "2018-07-01",
      "last modification": "2018-08-01",
      "owner": "<user-name>"
  }
}}
```

FIGURE 3. **An example of CouchDB JSON document from DoCA system.**

- Keypoint Localization: In this step low-contrast keypoints and edge keypoints are eliminated, what remains is strong interest points. The first is achieved by means of interpolation using quadratic Taylor series expansion of scale space.
- Orientation Assignment: An orientation histogram with 36 bins covering 360 degrees is created. Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak and local peaks within 80% of it are selected to assign multiple orientations to them which improves the stability of matching significantly.
- Keypoint Descriptor: Following the steps so far, each keypoint got image location, scale, and orientation assigned to it. Now, gradient magnitude and orientation are computed at each sample in a 16×16 neighborhood around the keypoint. Then divided into 4×4 sub-blocks by accumulating the contents as 8 bin orientation histograms. It is represented as a vector to form a keypoint descriptor.
- Keypoint Matching: Keypoints between two images are matched by identifying their nearest neighbors. Then a ratio of closest-distance to second-closest distance is taken, matches with a ratio greater than 0.8 are rejected.

D. AUDIO CONTENT ANALYSIS

inaSpeechSegmenter is a CNN-based audio segmentation toolkit [3] for studying gender equality based on men/female speech times in multimedia and it is reliable for large scale studies. We use inaSpeech with its original trained model because it performed well enough with Turkish speech, and the provided dataset was not big enough to allow training of a new model. inaSpeech works by first extracting a



I am the very model of a
modern Major-General,
I've information vegetable,
animal, and mineral,
I know the kings of England,
and I quote the fights
historical,
From Marathon to Waterloo, in
order categorical.

(a) Text version 1.

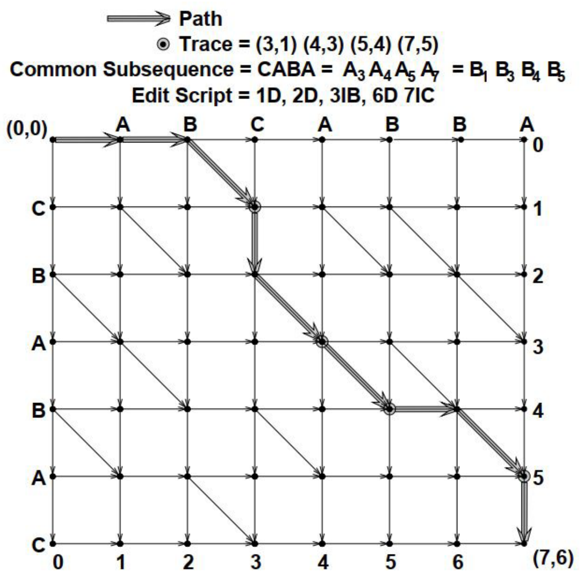
I am the very model of a
cartoon individual,
My animation's comical,
unusual, and whimsical,
I'm quite adept at funny gags,
comedic theory I have read,
From wicked puns and stupid
jokes to anvils that drop on
your head.

(b) Text version 2.

I am the very model of a ~~modern Major-General,¶~~
~~I've information vegetable, animal, and mineral,¶~~
~~I know the kings of England, and I quote the fights historical,¶~~
~~From Marathon to Waterloo, in order categorical~~cartoon individual,¶
My animation's comical, unusual, and whimsical,¶
I'm quite adept at funny gags, comedic theory I have read,¶
From wicked puns and stupid jokes to anvils that drop on your head.

(c) Resulting Diff.

FIGURE 4. **An example of how Google Diff output is visualized [57].**



Path
Trace = (3,1) (4,3) (5,4) (7,5)
Common Subsequence = CABA = A3 A4 A5 A7 = B1 B3 B4 B5
Edit Script = 1D, 2D, 3IB, 6D 7IC
(0,0)
A B C A B B A
C B A B A C
0 1 2 3 4 5 6
0 1 2 3 4 5
(7,6)

FIGURE 5. **An edit graph for two sequences A and B [9].**

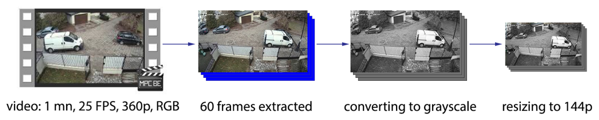


FIGURE 6. **Video pre-processing pipeline.**

Mel-Frequency Cepstrum (MFC) of the given audio using SIDEKIT [60]. Then it is fed through a CNN. The output is a segmentation of speech and music audio tracks, followed by another segmentation of speech tracks as female or male voices.

E. VIDEO CONTENT ANALYSIS

The aim of video content analysis in DoCA is to implement an efficient program that could determine whether a video is from a security camera or not. To achieve that we rely on the fact that security cameras are stationary, hence their footage should have more similar frames than normal videos. The approach relies on comparing consecutive frames from a video to determine a similarity ratio between them. Then based on a chosen threshold, any video can be classified as either security footage or not. The whole process follows three steps: pre-processing, computing a similarity ratio, decision.

In pre-processing, OpenCV is used to open the video and extract its metadata, namely resolution and FPS. FPS is used to determine which frames to use in the comparison because only one frame from each second is taken and then it is converted to grayscale. While the resolution is used to get the aspect ratio which is maintained while resizing the frame to only 144 pixels in height. This pre-processing as shown in Figure 6 allows for significantly faster computations and the use of fewer resources in the next step.

In the second step, a list of the previously processed frames is passed to a comparison function, where they are split into four different groups to allow multi-threading computation, each frame in each group is then compared to the next three frames, the similarity scores are averaged as one ratio.

Finally, this ratio is used to determine whether the video is security footage or not, the threshold can be hand tuned depending on the dataset to achieve better results. Structural Similarity Index (SSIM) is the method used in the comparison part. SSIM is a standard for image quality metrics because it solves the drawbacks of the simpler Mean Squared Error (MSE) [61], [62], by taking texture into account instead of being applied globally. Formally, SSIM is defined as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{1}$$

where μ_x, μ_y are the average of x,y respectively, σ_x^2, σ_y^2 are their variance, σ_{xy} is their covariance, $c_1 = (k_1\ L)^2, c_2 = (k_2\ L)^2$ are two variables to stabilize the division with weak denominator, L is the dynamic range of the pixel-values ($2^{bitsperpixel} - 1$), and $k_1, 2 << 1$. SSIM value is between 0 and 1, the higher the value the more similar the images.

F. FILE AND FOLDER WATCHDOG ANALYSIS

Part of documents analysis is tracking historical changes and storing different versions of a file over time. This can be easily done using *Watchdog* [45] a Python API library and shell utilities to monitor file system events. Whenever an event is detected, the following tasks are performed depending on the event type (creation, deletion, modification) and the file type:

- store the new version of old files in the database with their metadata. the new version is added as a child of the original file.
- auto-classify newly added files.

G. DOCUMENT CLASSIFICATION

As with any Natural Language Processing (NLP) project, this work starts by some pre-processing of text dataset, each document is stored as a string in an array containing all documents. The strings are then tokenized, cleaned from stop words and finally, we can perform stemming on our tokens. Then we use Gensim's Doc2Vec model [63] to vectorize the documents and store the resulting model for later use. Doc2Vec [64] allows unsupervised learning of continuous representations for larger blocks of text found in entire documents. To classify unlabeled data, two different approaches have been used, clustering using Affinity Propagation and Hierarchical clustering using a Linkage Matrix. The former is mentioned in section II-F. We chose Affinity Propagation because it eliminates the need for specifying the number of classes without being susceptible to noise like DBSCAN. AP has quadratic algorithmic complexity ($O(n^2)$) and uses the negative squared Euclidean distance between points, formally

TABLE 1. Document types and their numbers. * gifs are one frame, pdfs are multipage.

Document type-1	.doc	.docx	.odt	.ppt	.pptx	.odp	.xls	.xlsx	.ods
# of documents	5	11	5	8	13	5	12	9	4
Document type-2	.pdf *	.png	.jpeg	.jpg	.gif *				
# of documents	25	15	2	3	3				
Document type-3	.mp3	.wav	.mp4	.avi	.mpg				
# of documents	10	3	14	3	5				

TABLE 2. Experimental results.

Tasks	Time (s)	RAM usage (MB)	Accuracy
Document type-1 similarity analysis	22	421	-
Document type-1 diff analysis	1011	1158	-
Document type-1 classification	30	514	100%
Text search in document type-2	241	1312	100%
Image search in document type-2	324	7853	100%
Document type-2 classification	-	-	100%
Document type-3 audio content analysis	168	3183	93%
Document type-3 video content analysis	60	179	100%

TABLE 3. Comparison results.

Tasks	Time (s)	RAM usage (MB)	Accuracy
Document type-1 classification (DBSCAN)	24	630	91%
Image search in document type-2 (SURF)	452	6728	100%
Audio content analysis (pyAudioAnalysis)	105	1205	63%

defined as:

$$||a-b||_2^2 = \sum_i (a_i - b_i)^2 \tag{2}$$

The latter is a general family of clustering algorithms that build nested clusters. The hierarchy of these clusters is then represented as a Dendrogram i.e. a tree with its root representing the cluster that includes all samples, and each leave representing a single sample. We opted for an Agglomerative strategy; meaning that it is a "bottom-up" approach starting from the leaves and merging them in clusters successively. An implementation with a complexity of $O(n^2)$ is offered by scipy.cluster.hierarchy.linkage module [65] which we used with the following parameters: "Euclidean" for distance metric, and "average" for the method of clustering, this is also called the UPGMA algorithm (Unweighted Pair Group Method with Arithmetic Mean) [66] and defined as:

$$d(u,v) = \sum_{ij} \frac{d(u[i], v[j])}{|u| \cdot |v|} \tag{3}$$

for all points i and j where $|u|$ and $|v|$ are the cardinalities of clusters u and v, respectively.

IV. EXPERIMENTAL EVALUATION

A. EXPERIMENTAL SETUP

This project was implemented in Python 3 and tested on a local machine with the following specifications: CPU: Intel Core i7-7700HQ, RAM: 12 GB DDR4-2400, GPU: Nvidia GeForce GTX 950M 2GB GDDR5, OS: Ubuntu 18.04 LTS.

The codes are available at: https://github.com/husmen/DoCA_GUI/

B. DATASET INFORMATION

The dataset on which experiments have been done was provided by HAVELSAN [5] and includes the following file types with the number of files as shown in Table 1.

C. EXPERIMENTAL RESULTS AND DISCUSSION

Table 2 summarizes the results, it shows the running time, maximum RAM usage, and accuracy for each of the tests (omitted when unavailable). Text search in document type-2 results refers to the first run including OCR, following runs use the stored content directly from the database which makes them significantly faster. For image search in document type-2, we used a manually selected set of 11 logos to perform the search. The experiments show near perfect results in our use case.

D. COMPARISONS WITH OTHER METHODS AND TOOLS

Comparing the performance of our software to that of other researchers in a meaningful way is quite difficult due to the use of a non-standard dataset that our project was built around. However we can offer some insight with the results summarized in Table 3:

For Document type-1 classification, replacing Affinity propagation with DBScan resulted in an increase in speed (24ms) but also in RAM usage (630MB). Accuracy decreased to 91% due to noise.

For Image search in document type-2, replacing SIFT with SURF resulted in a decrease in RAM usage (6728MB) but it was significantly slower (452s).

For Audio classification, pyAudioAnalysis [67] library ran fast (105s) and used less memory (1205MB) than InaSpeech but at a heavy cost in accuracy with only 63%.

V. CONCLUSION AND FUTURE WORK

We achieved good and reliable results with reasonable running times and RAM usage which makes it a good framework for general document classification and analysis on different file types. It could serve as a starting point for future work on a more robust, flexible and feature-rich solution. The whole process can also be fully automated.

Future work will include some improvements and will address some of the current shortcomings: For feature matching, different and newer algorithms could be explored for better performance such as FLANN. The same thing goes for clustering, other methods like HDBSCAN can be explored [68]. As well as new vector distance metrics such as TS-SS [69]. Feature selection could also benefit from some improvements [70], [71]. For audio content analysis, different modules could be trained for each language because acoustic features of speaker gender are language dependent.

ACKNOWLEDGMENT

The authors are grateful to HAVELSAN Inc., for their mentoring at Open Innovation Competition on PARDUS File Classification and Analysis [5].

REFERENCES

[1] G. A. Bowen, "Document analysis as a qualitative research method," *Qualitative Res. J.*, vol. 9, no. 2, pp. 27–40, Aug. 2009.

[2] J. Salamon and J. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, Mar. 2016.

[3] D. Doukhan, J. Carrive, F. Vallet, A. Larcher, and S. Meignier, "An open-source speaker gender detection framework for monitoring gender equality," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2018, pp. 5214–5218.

[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. CVPR*, Columbus, OH, USA, Sep. 2014, pp. 1725–1732.

[5] *Open Innovation Competition on Pardus File Classification and Analysis*. Accessed: Sep. 30, 2018. [Online]. Available: https://inovasyon.havelsan.com.tr/havelsan/#/competition-open/4

[6] *Havelsan, a Turkish Software and Systems Company*. Accessed: Jan. 25, 2019. [Online]. Available: http://www.havelsan.com.tr/

[7] *Pardus, a Linux Distribution Developed With Support from the Turkish Government*. Accessed: Jan. 25, 2019. [Online]. Available: https://www.pardus.org.tr/

[8] *Google Diff Match Patch*. Accessed: Sep. 30, 2018. [Online]. Available: https://opensource.google.com/projects/diff-match-patch

[9] E. Myers, "AnO(ND) difference algorithm and its variations," *Algorithmica*, vol. 1, nos. 1–4, pp. 251–266, Nov. 1986.

[10] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *J. ACM*, vol. 24, no. 4, pp. 664–675, Oct. 1977.

[11] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Commun. ACM*, vol. 20, no. 5, pp. 350–353, 1977.

[12] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Sov. Phys.-Dokl.*, vol. 10, pp. 707–710, Feb. 1966.

[13] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[14] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Commun. ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[15] A. Andoni, R. Krauthgamer, and K. Onak, "Polylogarithmic approximation for edit distance and the asymmetric query complexity," in *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, Jul. 2010, pp. 244–252.

[16] S. Yang and R. Wei, "Tabdoc approach: An information fusion method to implement semantic interoperability between IoT devices and users," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1972–1986, Apr. 2019.

[17] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vis. Conf.*, Jul. 1988, pp. 147–151.

[18] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[20] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009, pp. 331–340.

[21] M. Muja, and D. G. Lowe, "Fast matching of binary features," in *Proc. 9th Conf. Comput. Robot Vis.*, May 2012, pp. 404–410.

[22] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.

[23] H. Ling and D. W. Jacobs, "Deformation invariant image matching," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, vol. 2, Oct. 2005, pp. 1466–1473.

[24] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[25] F. Moreno-Noguer, "Deformation and illumination invariant feature point descriptor," in *Proc. CVPR*, Jun. 2011, pp. 1593–1600.

[26] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "DaLI: Deformation and light invariant descriptor," *Int. J. Comput. Vis.*, vol. 115, no. 2, pp. 136–154, Nov. 2015.

[27] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[28] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Fundations Trends Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, Jan. 2008.

[29] L. Juan and O. Gwun, "A comparison of SIFT, PCA-SIFT and SURF," *Int. J. Image Process.*, vol. 3, no. 4, pp. 143–152, 2009.

[30] M. El-Gayar, H. Soliman, and N. Meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informat. J.*, vol. 14, no. 2, pp. 175–181, Jul. 2013.

[31] A. Lerch, *An Introduction to Audio Content Analysis: Application Signal Processing Music Information*. Hoboken, NJ, USA: Wiley, 2012.

[32] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Trans. Multimedia*, vol. 13, no. 2, pp. 303–319, Apr. 2011.

[33] Y. Qian, M. Bi, T. Tan, and K. Yu, "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 12, pp. 2263–2276, Dec. 2016.

[34] G. N. Meenakshi and P. K. Ghosh, "Automatic gender classification using the mel frequency cepstrum of neutral and whispered speech: A comparative study," in *Proc. 21st Nat. Conf. Commun. (NCC)*, Feb. 2015, pp. 1–6.

[35] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2015, pp. 4694–4702.

[36] D. Graham, S. Langroudi, C. Kanan, and D. Kudithipudil, "Convolutional drift networks for video classification," in *Proc. ICRC*, Nov. 2017, pp. 1–8.

[37] J. C. Nascimento and J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 761–774, Aug. 2006.

[38] Y.-L. Tian, R. Feris, H. Liu, A. Hampapur, and M.-T. Sun, "Robust detection of abandoned and removed objects in complex surveillance videos," *IEEE Trans. Syst., Man, C, (Appl. Rev.)*, vol. 41, no. 5, pp. 565–576, Sep. 2011.

[39] U. Contractor, C. Dixit, and D. Mahajan, "CNNs for surveillance footage scene classification," 2018, *arXiv:1809.02766*. [Online]. Available: https://arxiv.org/abs/1809.02766

[40] *Inotify-Tools on Github*. Accessed: Sep. 30, 2018. [Online]. Available: https://github.com/rvoicilas/inotify-tools

[41] *Listen Gem on Github*. [Online]. Available: https://github.com/guard/listen Accessed: Sep. 30, 2018.

[42] *Guard on Github*. Accessed: Sep. 30, 2018. [Online]. Available: https://github.com/guard/guard

[43] *Pyinotify on Github*. Accessed: Sep. 30, 2018. [Online]. Available: https://github.com/rvoicilas/inotify-tools
[44] *Pyfilesystem2 on Github*. Accessed: Sep. 30, 2018. [Online]. Available: https://github.com/PyFilesystem/pyfilesystem2
[45] *Watchdog on Github*. Accessed: Sep. 30, 2018. [Online]. Available: https://github.com/gorakhargosh/watchdog
[46] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[47] A. D'cunha and A. K. Sen, "Hierarchical approach for scientific document classification," in *Proc. Int. Conf. Comput., Commun. Autom.*, May 2015, pp. 100–104.
[48] A. Nidhi and V. Gupta, "Article: Recent trends in text classification techniques," *Int. J. Comput. Appl.*, vol. 35, no. 6, pp. 45–51, Dec. 2011.
[49] T. Xia and Y. Du, "Improve VSM text classification by title vector based document representation method," in *Proc. 6th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Aug. 2011, pp. 210–213.
[50] A. Díaz-Manríquez, A. B. Ríos-Alvarado, J. H. Barrón-Zambran, T. Y. Guerrero-Melendez, and J. C. Elizondo-Leal, "An automatic document classifier system based on genetic algorithm and taxonomy," *IEEE Access*, vol. 6, pp. 21552–21559, 2018.
[51] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. AAAI*, 1996, pp. 226–231.
[52] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
[53] J. Vlasblom and S. J. Wodak, "Markov clustering versus affinity propagation for the partitioning of protein interaction graphs," *BMC Bioinf.*, vol. 10, no. 1, p. 99, Dec. 2009.
[54] N. Kamel, I. Ouchen, and K. Baali, "A sampling-PSO-K-means algorithm for document clustering," *Adv. Intell. Syst. Comput.*, vol. 238, pp. 45–54, Jan. 2014.
[55] *Rdbms Dominate the Database Market, but NoSQL Systems are Catching Up*. Accessed: Sep. 30, 2018. [Online]. Available: https://db-engines.com/en/blog_post/23
[56] N. Leavitt, "Will NoSQL databases live up to their promise?" *IEEE Comput.*, vol. 43, no. 2, pp. 12–14, Feb. 2010.
[57] *Google Diff Match Patch - Diff Demo on Github*. Accessed: Jan. 25, 2019. [Online]. Available: https://github.com/google/diff-match-patch
[58] *Fuzzywuzzy Fuzzy String Matching in Python*. Accessed: Oct. 15, 2018. [Online]. Available: https://github.com/seatgeek/fuzzywuzzy
[59] A. P. Witkin, "Scale-space filtering," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 2, 1983, pp. 1019–1022.
[60] A. Larcher, K. A. Lee, and S. Meignier, "An extensible speaker identification sidekit in Python," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5059–5099.
[61] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[62] R. Dosselmann and X. D. Yang, "A comprehensive assessment of the structural similarity index," *Signal Image Video Process.*, vol. 5, no. 1, pp. 81–91, 2011.
[63] *Models.doc2vec Doc2vec Paragraph Embeddings, Gensim*. Accessed: Sep. 15, 2018. [Online]. Available: https://radimrehurek.com/gensim/models/doc2vec.html
[64] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," 2014, *arXiv:1405.4053*. [Online]. Available: https://arxiv.org/abs/1405.4053
[65] *Linkage, Scipy Hierarchical Clustering*. Accessed: Sep. 15, 2018. [Online]. Available: https://docs.scipy.org/doc/
[66] R. Sokal and C. Michener, "A statistical method for evaluating systematic relationships," *Univ. Kansas Sci. Bull.*, vol. 38, pp. 1409–1438, Mar. 1958.
[67] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015, Art. no. e0144610.
[68] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Germany: Springer, 2013, pp. 160–172.
[69] A. Heidarian and M. J. Dinneen, "A hybrid geometric approach for measuring similarity level among documents and document clustering," in *Proc. IEEE Second Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, Sep. 2016, pp. 142–151.
[70] J. Meng, H. Lin, and Y. Yu, "Two-stage feature selection for text classification," *Comput. Math. Appl.*, vol. 62, pp. 2793–2800, Oct. 2011.
[71] M. K. Elhadad, K. Badran, and G. I. Salama, "A novel approach for ontology-based dimensionality reduction for Web text document classification," in *Proc. IEEE/ACIS 16th Int. Conf. Comput. Inf. Sci. (ICIS)*, May 2017, pp. 373–378.

SÜLEYMAN EKEN received the M.S. and Ph.D. degrees in computer engineering from Kocaeli University, Turkey, where he has been a Research Assistant with the Computer Engineering Department, since 2010. His main research work focuses on distributed systems and big data analysis.

HOUSSEM MENHOUR is currently pursuing the B.Eng. degree in computer engineering with Kocaeli University. His main research work focuses on autonomous vehicles, deep learning, and digital document processing.

KÜBRA KÖKSAL is currently pursuing the B.Eng. degree in computer engineering with Kocaeli University. Her main research work focuses on deep learning, digital document processing, and file systems.