NPS-CS-22-002

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

EXPEDITIONARY DOMAIN AWARENESS – INTELLIGENCE SUPPORT TO NECC & NECC SUPPORT TO INTELLIGENCE ANALYSIS (NECC FOCUS)

by

Arijit Das

October 2022

Prepared for: N2/N6 Information Warfare, Naval Intelligence.
This research is supported by funding from the Naval Postgraduate School, Naval Research Program (PE 0605853N/2098). NRP Project ID: NPS-22-N270-A

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
October 2022	Technical Report	10/24/2021	10/22/2022

3. TITLE AND SUBTITLE

Expeditionary Domain Awareness - Intelligence Support to NECC & NECC Support to Intelligence Analysis (NECC FOCUS)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
		0605853N/2098
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
NPS-22-N270-A; W2223		

6. AUTHOR(S)

Arijit Das

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Naval Postgraduate School Monterey, CA	NPS-CS-22-002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Naval Postgraduate School, Naval Research Program; N2/N6 Information Warfare, Naval Intelligence, Mr. Robert Inscore	NRP; N2/N6	NPS-CS-22-002; NPS-22-N270-A

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Navy Expeditionary Combat Command (NECC) community gathers information and intelligence documents that accumulate over time on file stores. The intelligence consumers needed a method to search all prior knowledge documents, preferably based on common language keywords and phrases. This challenge could be solved by working with an existing vendor product with the associated licensing, support, and maintenance. The Naval Postgraduate School (NPS) team took a computer science (CS) approach to identify the various workings of a document store/search portal (system). An evaluation of each technology step involved was conducted, and potential solutions and their associated costs were considered. The team found that given user specifications, a tech-savvy team, and combined with open-source and Department of Defense (DOD)–licensed software, one can build and maintain a system that meets the requirements of the Department of Navy (DON) community.

15. SUBJECT TERMS

Naval Expeditionary Combat Command, NECC, Expeditionary Domain Awareness, intelligence, operations, collaboration, portal, information stream, Naval Expeditionary Combat Forces, Tribes, database, Hadoop, artificial intelligence, Machine Learning.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	U	35

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code*
Arijit Das	(831) 402 9187

STANDARD FORM 298 (REV. 5/2020)

Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau
President

Scott Gartner
Provost

The report entitled "Expeditionary Domain Awareness - Intelligence Support to NECC & NECC support to Intelligence Analysis (NECC focus)" was prepared for N2/N6 Information Warfare, Naval Intelligence and funded by Naval Postgraduate School, Naval Research Program (PE 0605853N/2098).

Distribution Statement A: Approved for public release. Distribution is unlimited.

This report was prepared by:

Arijit Das
Research Associate

Reviewed by:

Gurminder Singh, Chair
Computer Science

Released by:

Kevin B. Smith
Vice Provost for Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Navy Expeditionary Combat Command (NECC) community gathers information and intelligence documents that accumulate over time on file stores. The intelligence consumers needed a method to search all prior knowledge documents, preferably based on common language keywords and phrases. This challenge could be solved by working with an existing vendor product with the associated licensing, support, and maintenance. The Naval Postgraduate School (NPS) team took a computer science (CS) approach to identify the various workings of a document store/search portal (system). An evaluation of each technology step involved was conducted, and potential solutions and their associated costs were considered. The team found that given user specifications, a tech-savvy team, and combined with open-source and Department of Defense (DOD)–licensed software, one can build and maintain a system that meets the requirements of the Department of Navy (DON) community.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The Navy Expeditionary Combat Command (NECC) community gathers a substantial quantity of intelligence via datasets that are processed and summarized into reports. These reports accumulate over time on disk stores, distributed across multiple computers. This poses a problem when searching for historic documents. Archiving the documents to a single server, accessible to the entire NECC community, would enable and enrich ongoing and future intelligence analysis. Further, documents in a single database store can be preprocessed and analytics gathered for matching with keyword search.

The initial understanding was that the NPS team would be provided raw datasets to evaluate and analyze. After several meetings with NECC, it became clear that data is preprocessed and summarized in the form of reports. Reports can be in any format, namely Adobe Acrobat PDF, Microsoft Word and Microsoft PowerPoint, various image types and plain text. These reports are distributed to the community via email/file systems and need to be searched later. There is no centralized system to store, analyze, and generate analytics (based on the documents) for the community.

The team took a Computer Science (CS) approach to understand the challenges and evaluate a solution that could be built with in-house developers and Department of Defense (DOD) licensed software. NPS researchers studied an industry standard, IBM Content Management Systems (CMS). The team had prior background with processing large datasets and extracting analytics using the Hadoop Distributed File System (HDFS) and a relational database. Common algorithms/code focus on plain text; since the NPS team was familiar with processing binary files and extracting needed information in plain text, this could be applied to the non-plain text documents. Data growth is an important consideration that should be handled seamlessly with technology. For this the NPS team used its background in Big Data technologies with HDFS.

After documents were loaded into the system, algorithms had to be researched that could extract the keywords in plain text and create metrics. These metrics will aid in generating intelligent results when the user community searches historical information with keywords and phrases. The NPS team used its background in document classification to evaluate existing industry standard algorithms that may be applied to this problem set.

For any such system to be viable, the user community needs a friendly user interface. There is also the challenge of multiple devices like a laptop, desktop, handheld devices, and smart phones. The NPS team looked at openly available technologies like HTML5, JavaScript, open-source webserver, and Python programming libraries. The frontend (browser on laptop/phone) needs to send data over the internet to a webserver (middleware) that is subsequently sent to the database (Oracle) backend. For all of this to work, the three parts need to be compatible.

Overall, technologies need to be available via DOD licensing and be cost effective. The plan was not to recommend any esoteric or custom software that might be a financial challenge and face a lack of developer community support. Instead of total reliance on vendor consulting teams, these technologies must be supported by DON in-house technology teams with training and minimal vendor support. A basic architecture is proposed that would enable end users to load the documents into a single database store, preprocessed for analytics and searched using keywords.

B. REPORT ORGANIZATION

The content management section discusses the industry standard tool that was selected and evaluated by the NPS computer science team to model a cost-effective and supportable solution that could fit the NECC needs. The technology section evaluates the various components that goes into putting an overall system together, step by step. The findings section discusses the challenges and lessons learned. Finally, recommendations and future work are summarized.

II. INDUSTRY SOLUTION

A. CONTENT MANAGEMENT SYSTEM

Gartner is a technology research company that ranks software based on criteria relevant to end-user community. If one were to buy a CMS application, they would first refer to Gartner reports for guidance. For CMS software Gartner ranks IBM high along with other industry vendors. The NPS team members had prior familiarity with IBM software and thus chose IBM CMS for this study. The following diagram depicts 3 of the key steps in the document management process from the IBM CMS: Capture, Transform, and Deliver.

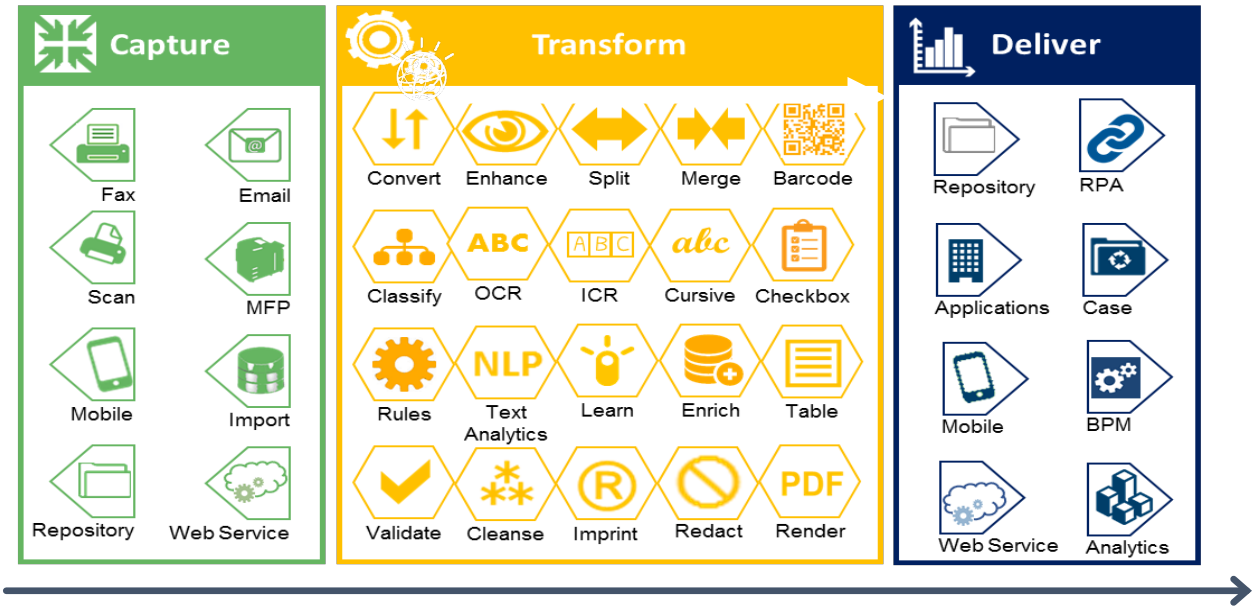


Figure 1. IBM Content management

The NECC intelligence documents come in many formats including PDF, Microsoft Word, Microsoft PowerPoint, various image types and plain text. Keywords need to be extracted for classification and searching. Extracting keywords from binary format files is done by first converting the document to an image during the Capture phase.

Optical Character Recognition (OCR) is next used to extract keywords. The extracted words need to be cleaned and fed to a classification algorithm to create metrics. These metrics are used when documents are searched using keywords (query). This is the Transform phase. Finally, during the Deliver phase the results are consumed, via a variety of potential devices, by the end users. The NPS Team learned from this model and replicated the needed functionality using open-source and DOD licensed software. The study also considered that the IBM CMS licensing model contained additional features, unnecessary to the NECC community. And as such, applicable components of the system were studied and evaluated using alternate software packages.

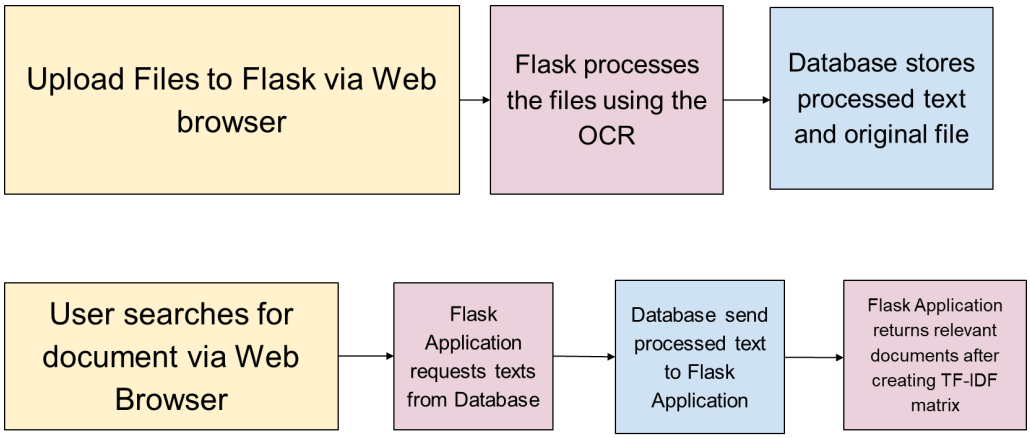


Figure 2. 3-Tier architecture (NPS work).

Any system has three basic requirements, a backend database to store all the files and keywords, a web/application server-based platform (middleware) to run the algorithms for capture, extract, and classification, and finally a browser-based method for consuming the information (frontend). The Flask webserver receives the data from the browser and does preprocessing on the Flask server using Python code (middleware) and handles communication with the database. A complete system consists of backend, middleware and frontend communicating with each other using network protocols. This is known as a 3-Tier architecture, and it can be implemented using a variety of programming languages including Python (well supported by the community).

The Flask middleware implemented in this project is a Python based framework that and enables one to run all the code that does the processing. The backend is an Oracle database that can handle a variety of data types (text, binary). The user interface is a browser, accessed using a laptops/desktops or other smart device. To study the phone interface an Android device was used along with a Java to Python container to run the Python programs.

III. TECHNOLOGY

A. INTRODUCTION

The system architecture involves several steps, and they are discussed in this section. Each step involves different software packages. The team chose the Python programming language as it is a widely used and supported platform. The functional steps are put together using a Python web/application server FLASK (middleware).

B. DOCUMENT TO IMAGES

The first step is to convert the documents to images. The team chose to use the Python Pillow Library (PIL). PIL is a collection of Python code to handle reading of documents and converting them to images (JPEG). The diagram below explains the flow.

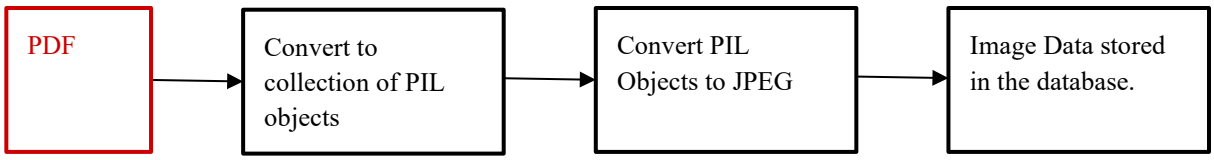


Figure 3. Documents to Images using the PIL library.

C. IMAGE TO TEXT

One of the most common methods for extracting words from images includes a specific type of Computer Vision model called an Optical Character Recognition (OCR) model. Computer Vision refers to computational algorithms that can identify features within an image to provide analysis of what is displayed in the image. By studying the differences in color between pixels in an image, these computer algorithms can identify edges and curves. As a result, these algorithms can be used for a variety of applications, including identifying text in images. This specific study utilized OpenCV, a series of computational methods meant to edit and process images.

The diagram shows how an OCR system functions internally at a high level.

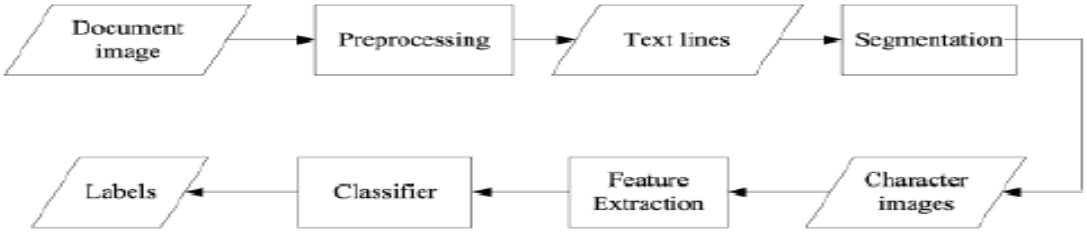


Figure 4. Basic workings of an OCR algorithm.

For the OCR model, this study utilized the pre-trained Tesseract OCR model created by Google. A pre-trained model refers to algorithms that have already been given data to help them identify patterns within this data. Without pre-trained OCR models, data scientists have to find large quantities of images, with each letter in different orientations and fonts to set up the OCR model, before the model can recognize the text in license plates or other images. Tesseract algorithms must be configured for the specific text for the project, and it was done using custom regular expressions (regex).

After OCR extracts the words, the text must be cleaned using the Python Natural Language toolkit (nltk). The punctuation removal, capitalization normalization, stop-word elimination and lemmatization are done using nltk.

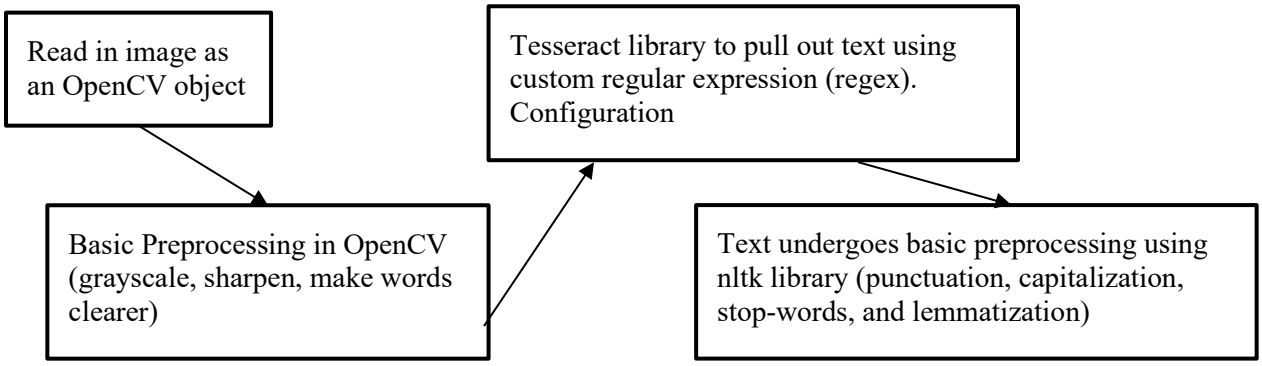
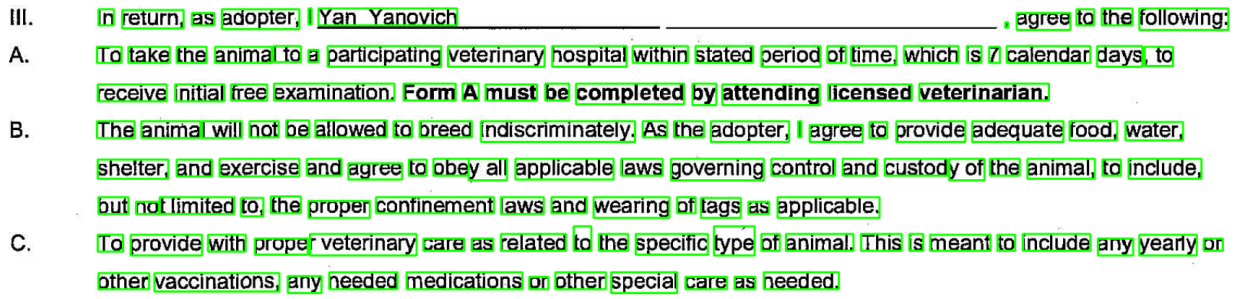


Figure 5. Using the trained OCR model Tesseract and natural language toolkit nltk

The Tesseract OCR library was selected because it can recognize the text lines within an image, which helps the final model recognize skewed images. One of the benefits of using a pre-trained model, such as the Tesseract OCR library, is that we do not need to de-skew, or straighten, images during pre-processing and still retain the image quality. However, this model is not optimal and can be further improved by performing basic preprocessing techniques using OpenCV, an open-source image recognition library. The image can be cleaner if we convert the images to grayscale, or black and white, and increase the contrast between the shadows and highlights, thereby also improving the OCR's accuracy.


III. In return, as adopter, I Yan_Yanovich _____ _____ . agree to the following:

A. To take the animal to a participating veterinary hospital within stated period of time, which is 7 calendar days, to receive initial free examination. **Form A must be completed by attending licensed veterinarian.**

B. The animal will not be allowed to breed indiscriminately. As the adopter, I agree to provide adequate food, water, shelter, and exercise and agree to obey all applicable laws governing control and custody of the animal, to include, but not limited to, the proper confinement laws and wearing of tags as applicable.

C. To provide with proper veterinary care as related to the specific type of animal. This is meant to include any yearly or other vaccinations, any needed medications or other special care as needed.

Figure 6. Example of the text recognized by the OCR (sample data).

The OCR algorithm distinguishes the words from text and encloses them in a green box, which are then stored as plain text as shown below.

```
HL In return, as adopter, | Yan_Yanovich , agree to the following:

A. To take the animal to a participating veterinary hospital within stated period of time, which is 7 calendar days, to
receive initial free examination. Form A must be completed by attending licensed veterinarian.

B. The animal will not be allowed to breed indiscriminately. As the adopter, | agree to provide adequate food, water,
shelter, and exercise and agree to obey all applicable laws governing control and custody of the animal, to include,
but not limited to, the proper confinement laws and wearing of tags as applicable.

C. To provide with proper veterinary care as related to the specific type of animal. This is meant to include any yearly or
other vaccinations, any needed medications or other special care as needed.
```

Figure 7. Extracted text data from the OCR

As shown in the above figures, the Tesseract OCR picks up almost all the words in an image or document, thereby allowing it to gather an extremely close replica of the file contents. To use the Tesseract OCR and OpenCV libraries, each page of the PDF must be converted into an image. The text from each page is then combined to create the final raw text from the files. After evaluating the OCR and other Python libraries to pull out all the

relevant text from the document, the various preprocessing methods for natural language processing (NLP) were studied.

Preprocessing is an extremely important part of working with natural languages. All natural languages, such as English, are geared towards communicating smoothly and effectively with other humans. However, for a computer that only understands numbers, there is unnecessary data that does not provide important information.

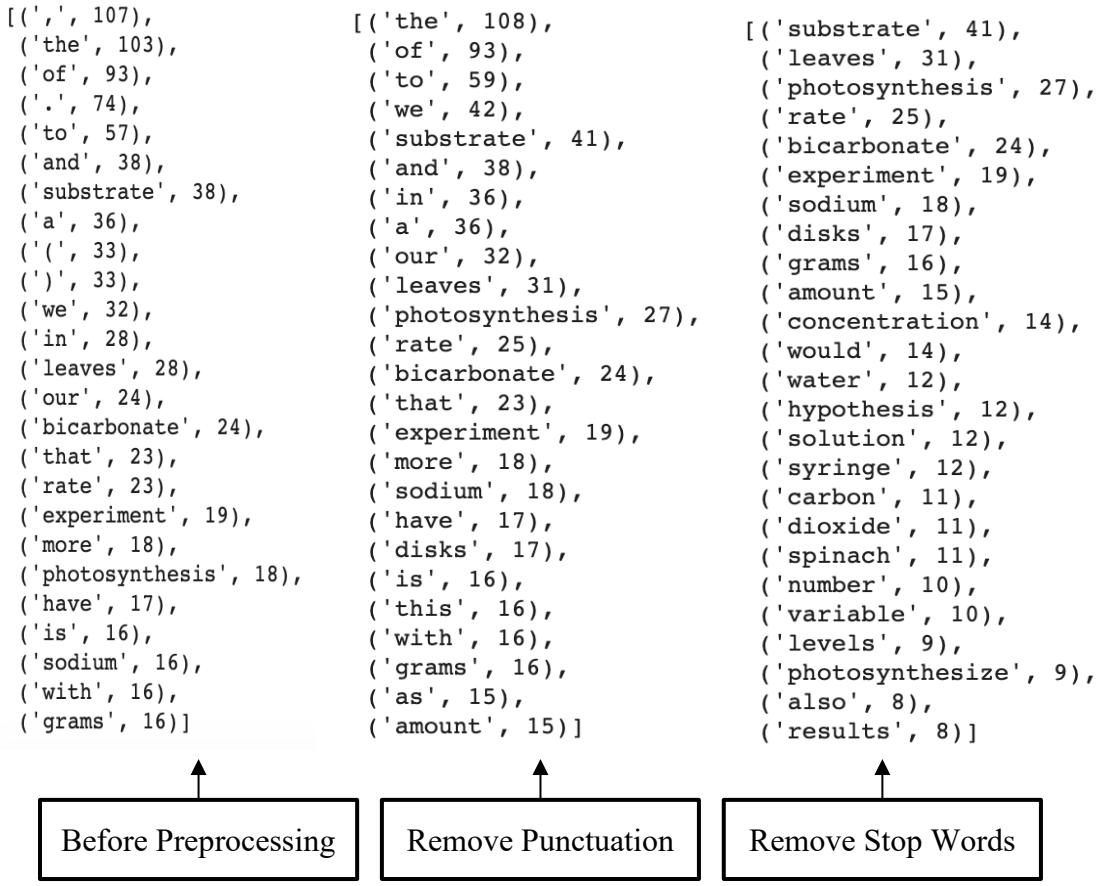


Figure 8. Word frequency distributions at each preprocessing stage.

The initial preprocessing consisted of removing punctuation, capitalization, and tokenizing the text into a list of words. Afterward, we removed stop words or words in natural language, which do not provide meaning or context, and lemmatized the words, thereby converting them to their root word. Finally, we calculated the word frequency distribution to see what content was left after preprocessing.

After removing punctuation and capitalization, the frequency distribution changes, but still returns words that are not unique and relevant to the document. After removing these stop-words and lemmatizing the data, we can see the distribution change drastically to include words pertaining to the document's topic, which helps to improve the accuracy of our search algorithm later.

D. TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY (TF-IDF)

Many studies have been conducted to test the efficacy of different types of document search algorithms. Document search algorithms can be defined as computational algorithms that can search through a series of documents and return relevant documents based on a query or a search phrase.

One of the most efficient and popular methods of sorting through documents is by using Term Frequency - Inverse Document Frequency vectors (TF-IDF) (Qaiser et al, 2018). TF-IDF vectors are numbers that represent how relevant a specific word is to a document in a collection of documents. These numbers are calculated by comparing the frequency of a term to the frequency of the same term in other documents, as shown in the following equation:

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{\mathrm{N}}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j
df_i = total number of documents (speeches) containing i
N = total number of documents (speeches)

Figure 9. Equation for calculating TF-IDF scores (Siddiqui, 2019).

The frequency of a word can be calculated by counting the number of times the term appears in the document compared to the total number of words in that document. Then, comparing this frequency to the frequency of the same term in other documents gives us our TF-IDF value. Next, by calculating the TF-IDF value for every word in all the documents, we can determine which keywords are the most relevant to each document. Next, using these keywords, we can construct an algorithm that will accurately return relevant documents. Finally, when a user enters a search phrase or query, documents with keywords matching the search phrase are returned.

E. COMBINING TF-IDF AND OCR

By combining these two technological concepts of TF-IDF and OCR, we hypothesized that the OCR model can significantly improve the accuracy of a TF-IDF-based document search engine, specifically when working with paper documents.

F. COSINE SIMILARITY

After creating the matrix with all the TF-IDF vectors, we must calculate the similarity of each document vector to the query vector (keywords). Each vector is represented in a 3D space, such as the one depicted in figure 10 below. The closer the vectors are, the more related they are to each other.

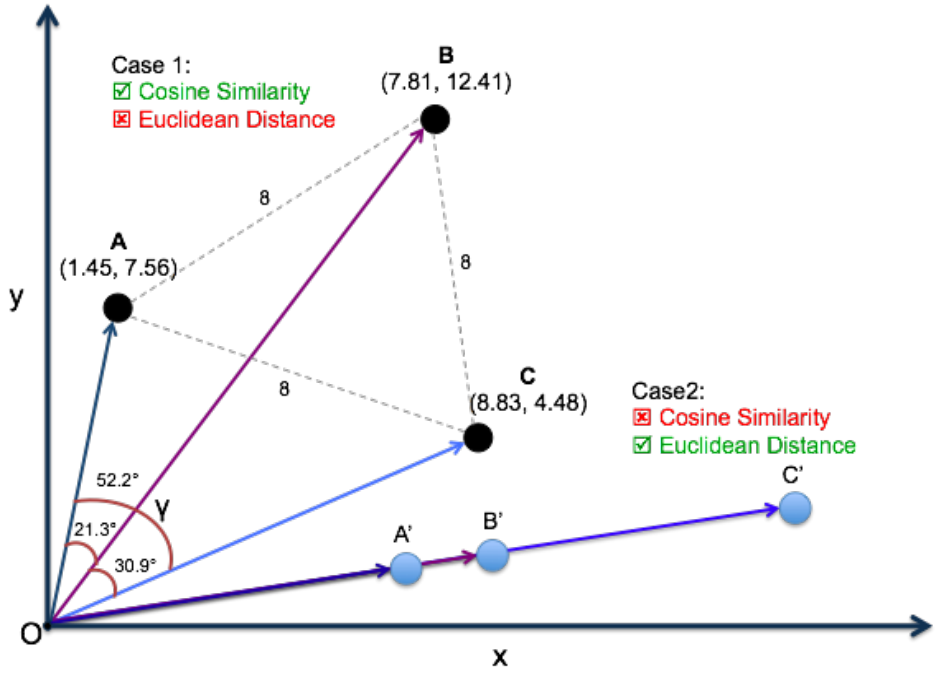


Figure 10. Cosine Similarity (Nagella, 2019).

Two major ways to calculate closeness exist. These are (i) Euclidean distance and (ii) Cosine similarity. Euclidean distance measures the magnitude and angle of the vectors while cosine similarity measures only the angle between the vectors (Nagella, 2019). Euclidean distance is useful for exact sentence matches, where the vectors are likely to be angled the same way but with varying magnitudes, such as the vectors depicted in case 2 of Figure 7. However, since we are trying to match long documents to short queries, the magnitude would not be similar, resulting in low precision and accuracy when returning related documents. We calculated the cosine similarity to find the documents with the smallest angle between the query and document.

By focusing on the last row in the cosine similarity matrix, we can see the similarity scores for the documents to the query, and in turn, return the document with the highest score. It is important to note that the last element in the last row does not count as a relevant document since it represents the similarity when comparing the query to itself. Figure 11 provides an example of a cosine similarity matrix for eight documents and a single query. The matrix is very similar to a confusion matrix.

```
[[1.         0.03005134 0.01296408 0.         0.         0.03133414 0.03445396 0.03208275 0.44255733]
 [0.03005134 1.         0.00562574 0.         0.         0.03673798 0.08287585 0.06649821 0.        ]
 [0.01296408 0.00562574 1.         0.         0.         0.02335712 0.02277339 0.01615491 0.        ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.        ]
 [0.03133414 0.03673798 0.02335712 0.         0.         1.         0.1927276  0.02965571 0.        ]
 [0.03445396 0.08287585 0.02277339 0.         0.         0.1927276  1.         0.04533234 0.        ]
 [0.03208275 0.06649821 0.01615491 0.         0.         0.02965571 0.04533234 1.         0.        ]
 [0.44255733 0.         0.         0.         0.         0.         0.         0.         1.        ]]
```

Figure 11. Cosine Similarity Matrix (8 documents and 1 query)

The matrix shows the similarity numbers between 8 documents and the query which is set of 1 or more keywords. The higher numbers mean the matching is better and that document is selected.

G. MIDDLEWARE AND DATABASE SETUP

To store all the documents for the finalized product, we used an Oracle XE Database, which is a laptop version. Oracle databases scale from laptop computers (less computing power) to large servers (greater computing power), thus testing on the XE provided a good base line understanding. Figure 12 (ER diagram) shows a schema of 3 tables. The central table is the Filenames table that has unique IDs for each file. The Files table contains the binary version of the file stored as a database Blob type. The text table has all related text for the file. All tables are linked via the File ID to maintain data integrity.

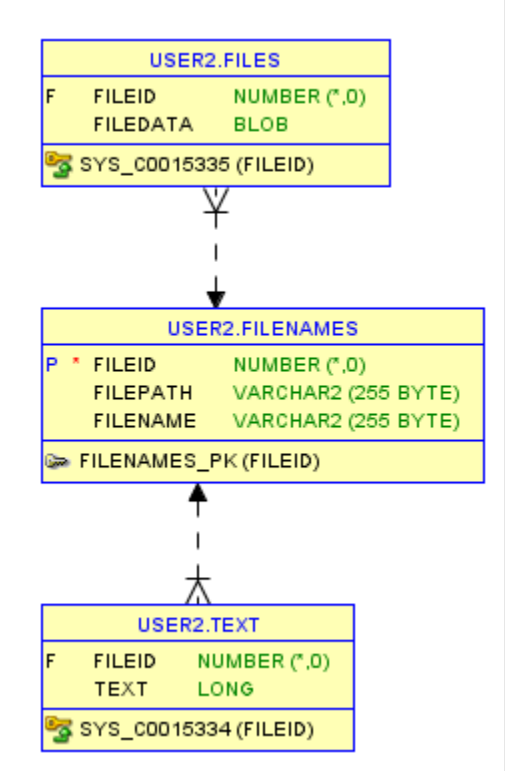


Figure 12. Database Schema

For each uploaded file, a new file ID is generated to easily access the data and connect the two tables together.

The figure below summarizes the 2 key flows of information in the user experience. Initially the documents are loaded using the browser interface on the end-user device, following which the Flask applications goes through the steps of capture, preprocessing and store. Subsequently the documents are searched with queries (keywords) and the best match document returned. The load and the search actions are independent of each other.

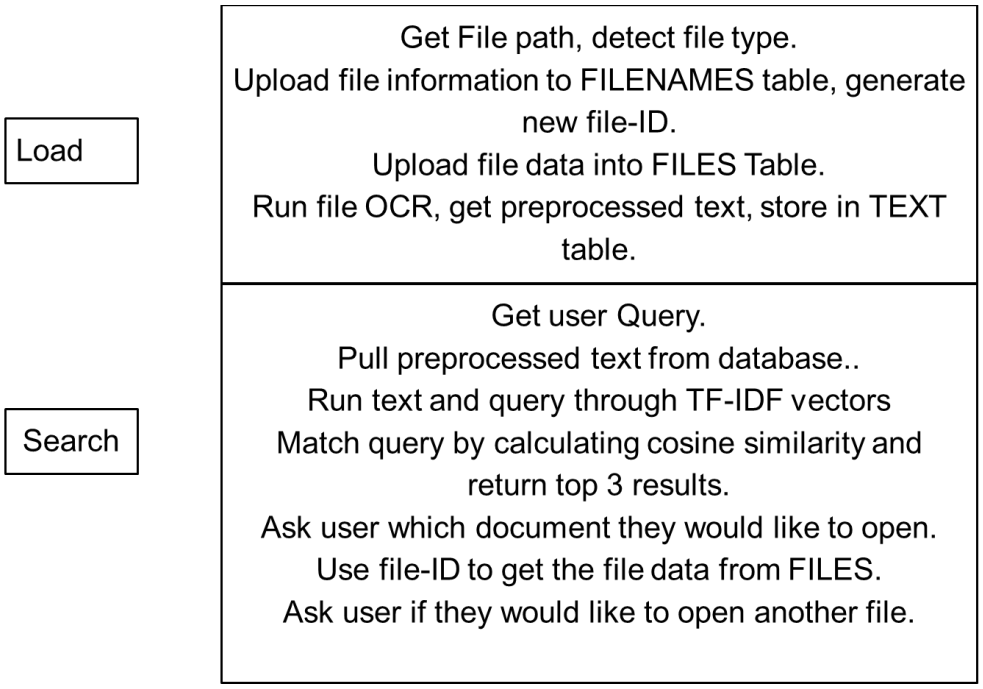


Figure 13. Document Load process and Search steps.

Each time a file is loaded, it goes through the OCR process followed by database store. The search will access the stored information to find the best match.

H. END USER EXPERIENCE

A Flask framework allows a browser with HTML/CSS frontend to connect to a Python backend. Flask applications are deployed as web applications easily accessible using a smart phone or laptop. The Flask framework automatically allows the user to search for an existing file and interface with the backend code to load it into the Oracle database. The figure below shows the same interface on different devices. A custom phone app was not chosen as the Android device uses a Java/Kotlin codebase which is not compatible with the Python libraries. Thus, using the browser option makes it a web-based application that works on both phones and laptop/desktops.

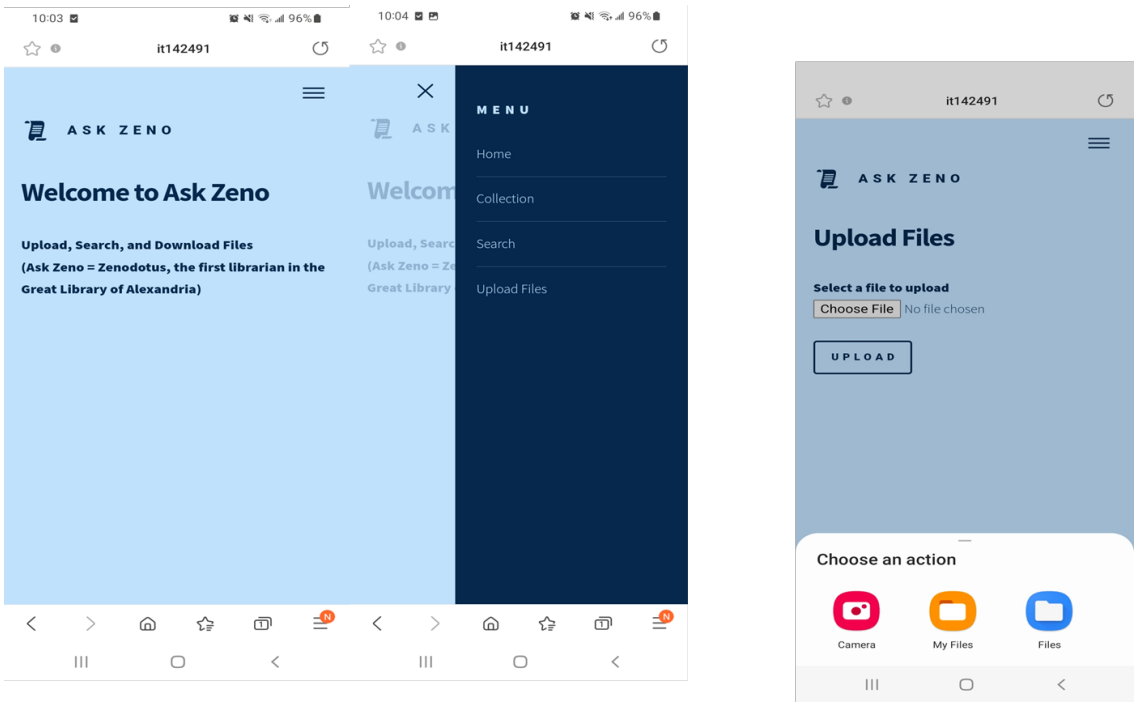


Figure 14. User interface on different devices

One way to integrate Python code into Java is to utilize a Java library called Jython. This library allows users to run Python code in a Java compiler. This option would entail a larger coding effort.

I. TECHNOLOGY CONCLUSION

The open-source community and the DoD/DoN licensing mechanism provides many options to achieve technology projects. A user community working with a technology savvy team can research the software and integrate it into a viable product.

IV. FINDINGS AND CHALLENGES

The NPS team started building a sandbox to evaluate the possible technologies. For a document store that can scale up, an Oracle vendor database product was considered. Using the Oracle XE laptop version, the team loaded documents and wrote SQL search queries. In the sandbox, NPS used the laptop version (database) with the assumption that a production Oracle database will work with the same codebase as the laptop version.

In the studied system, keywords need to be extracted from the documents. While it is easy to achieve this with plain text, with binary format documents, the solution is to use optical character recognition (OCR) technology. The first step is to convert the documents to image format and then use the OCR application to extract the keywords. Extracted keywords need to be cleaned of punctuation marks and stop-words (words used for grammatical sense) and lemmatized (variations of a word need to be made one). All final extracted words are stored along with the original documents, thus the database handles binary and plain text datatypes.

For each document loaded, the keywords are employed to create a matrix using term frequency-inverse document frequency (TF-IDF) vector algorithms. To calculate distance metrics, the cosine similarity algorithms are run on the matrix. Distance metrics are critical when the end users search for documents using keywords and phrases, as they will help generate a list of documents that are closest to the search string.

The team evaluated the frontend on a laptop using a browser. The middleware is from Flask (open-source application server), which is a Python programming language product. Flask lets one build the full software application in Python, so all the algorithms are Python packages that can be deployed to the Flask webserver and use the database as a store. When the frontend is deployed on an Android phone, it uses the Java programming language while Flask uses Python. A workaround is to use a Java to Python connector Jython, which allows Java applications to use Python libraries/code. This is an

extra layer of software that can increase execution time and degrade the speed of execution as data grows.

The system studied by the team requires that each time a document is loaded, all the calculations must be redone; this can be a challenge when the number of documents starts to grow. The sandbox did not fully test data growth using a HDFSs system.

Initial NECC results were encouraging. The study has aided in raising awareness of the problem. Follow-on research needs to be conducted before this solution can be implemented into production.

V. FUTURE WORK

The NPS team studied the document store/search system on a laptop sandbox, so the next step would be to scale up the evaluation. A server-based system can be used with a HDFS backend to understand the challenges of large-volume execution. A repository of datasets in the terabyte range will be a more realistic test of the system. Middleware technology needs to work on all platforms; if it works with Python and not with Java then a more generic middleware architecture needs to be researched and evaluated. Frontend technologies need to be examined on a wide range of devices, with large user community involvement. The middleware architecture needs to handle user growth for loading/searching of documents, thus more options beyond Flask need to be evaluated.

Additionally, there are many DOD CMS vendors who can be reached to present their solutions and evaluated. More studies need to be done with other DOD entities that may have already solved this problem.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Qaiser, S and Ali, R, July 2018, https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents

Siddiqui, S, January 2019, https://medium.com/shallow-thoughts-about-deep-learning/can-tfidf-be-applied-to-scene-interpretation-140be2879b1b

Nagella, V. S, December 2019, https://medium.com/@sasi24/cosine-similarity-vs-euclidean-distance-e5d9a9375fc8

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943