A Toolkit for Text Extraction and Analysis for Natural Language Processing Tasks

Tshephisho Joseph Sefara
Data Science Department
Council for Scientific and Industrial Research
Pretoria, South Africa
tsefara@csir.co.za

Mahlatse Mbooi
Data Science Department
Council for Scientific and Industrial Research
Pretoria, South Africa
mratsoma@csir.co.za

Katlego Mashile
Data Science Department
Council for Scientific and Industrial Research
Pretoria, South Africa
jmashile@csir.co.za

Thompho Rambuda
Data Science Department
Council for Scientific and Industrial Research
Pretoria, South Africa
trambuda@csir.co.za

Mapitsi Rangata
Data Science Department
Council for Scientific and Industrial Research
Pretoria, South Africa
mrangata@csir.co.za

***Abstract*—Text extraction is an important part of natural language processing (NLP) tasks. Most NLP tasks like text classification, machine translation, text-to-speech, text-based language identification, text summarization, and named-entity recognition involve the use of textual data. Such data is limited for low-resourced languages making it difficult to experiment advanced NLP techniques on these languages. This paper presents a Python-based toolkit for text analysis and text extraction from different types of images, documents, and audio files. The toolkit is built as a library that has functions that can be imported and utilized for text extraction.**

Keywords—natural language processing, text extraction, text analysis

I. Introduction

With the rise of digital age, there is high number of data existing in various format. This data can be in structured or unstructured format. Much of the data is in unstructured format and it is a challenging task for data extraction processes. Data extraction is a process of extracting data from a source file into a format that is suitable for a task being solved.

Data is an important part of natural language processing (NLP) since most NLP tasks are data driven. Data exists in different forms such as such as text, images, web pages, audio, sensors and many more. Some of these files are not in a current format that can be used in NLP tasks. Different software programs are required to process each type of file to extract the contents.

Data extraction plays a vital role for low-resourced language since most of the data exists in a format that cannot be used. Low-resourced languages are languages that have a very limited data that can be used for NLP problems. This toolkit slightly enables access to the data set found in unstructured files. Fortunately, the data can now be extracted from those unstructured files, apply preprocessing tasks, and used for NLP tasks such as topic modelling and text similarity.

Topic modelling is a process of learning, recognizing, and extracting topics from a collection of documents. These documents can be in any language. There are different types of topic modelling algorithms, namely, Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) [10], and many more. Topic modelling enable a user to know the topic(s) of a group of documents. On the other hand, text similarity is a task of identifying similar documents using identical contents to measure the similarity score. Text similarity is used in text summarization, machine translation, question answer session, topic detection, text categorization, clustering, and information retrieval [14].

This paper proposes the implementation of text extraction and text analysis toolkit that can also be used as a library that can be embedded or imported in other projects. The toolkit can work in any language (limited functionality) but it was tested on English for all its functions.

This paper is organized as follows: Section II discusses the background and literature study. Section III discusses the methods used for text extraction and text analysis. Section IV discusses the toolkit' methods and how these methods can be used while Section V concludes with future work.

II. Background

A. Data Extraction Methods

Data extraction methods depend on whether the data source is structured or unstructured. Structured data sources adhere to a certain format whereas unstructured data sources may contain text, images, web pages, audio and many more. We give a review of some of the data extraction libraries.

Pandas [1] is an open-source Python library. It is a BSD-licensed library, developed in 2008 by Wes McKinney. The Pandas package provides numerous tools for data analysis and multiple data structures that can be used for data manipulation tasks. It is a powerful input/output (IO) system for loading data from a wide variety of file formats or data sources through the "read" function, which loads the data into a data frame. The Pandas library has several advantages. For starters, it presents data in a form that is suitable for data analysis. Additionally, it has methods for data filtering. Finally, Pandas extracts data from a variety of formats such as Comma Separated Values (CSV), JavaScript Object Notation (JSON), and Excel file formats such as Microsoft

Excel Spreadsheet (XLS) and Microsoft Excel Open XML Spreadsheet (XLSX).

Web scraping helps in converting unstructured data into a structured data that can be utilized for extracting insights. While Extensible Markup Language (XML), and Hyper-Text Markup Language (HTML) are both markup languages that can be extracted using Pandas, the files may occasionally be too complex. Numerous libraries have been developed for the purpose of extracting data from web pages. Well known extraction libraries for web pages in Python include Beautiful Soup [18], selenium [16], and scrapy [17].

Beautiful Soup [18] is a Python library widely used for web scraping. Leonard Richardson developed the library in 2004 under an MIT license called BeautifulSoup3 which was discarded in 2020 for the latest version called BeautifulSoup4 in 2021. It creates a parse tree for parsing HTML and XML documents. Beautiful Soup can easily be combined with other parsers like lxml [19]. It also works well with poorly designed HTML and has numerous functions, making it the most widely used web scraping tool. Apart from HTML and XML files, Beautiful Soup can also be used to extract data from electronic publication (EPUB) file formats.

Text files are text-based files that can be classified as plain text files or rich text files. Plain text files have no special formatting and are full of text, whereas rich text files may contain non-text content such as images and some formatting. Plain text file formats include text (TXT), and rich text file formats include Rich Text Format (RTF), Microsoft Word (Doc), Microsoft Word Open XML Format Document (Docx), Open document format (ODF) formats such as OpenDocument Text (ODT) and OpenDocument Text Template (OTT), and Portable Document File (PDF).

Microsoft Word documents such as doc and DOCX are frequently used for text-based data. DOC is the predecessor of word and DOCX is the latest version of word. These file formats may contain in-line addition of tables, images, hyperlinks, etc. DOC file types can be extracted using the TextExtract library [36]. Additionally, it can be used to extract text from images. TextExtact is licensed under the MIT license, authored by Sayar Mendis.

Numerous data extraction libraries are available for the latest version of Word, DOCX. The Doc2txt library scrapes text and images from word documents. Additionally, it enables for the extraction of text from images. It is authored by Ankush Shah [20].

ODF is an XML-based file format similar to DOCX file formats. ODF formats can be extracted using ODFPY library. ODFPY is licensed under the Apache software license, authored by Soren Roug [21].

PDF format is one of the most widely used file format for text and graphic documents. PDF file format is the most unstructured file format and extracting data from such file formats is a complicated task. Pdfminer.six library [22] can extract text from PDF files it is built from from PDFMiner software authored by Yusuke Shinyama.

Postscript (PS) file formats can be extracted using Python. It iterates over all the lines in the file and extract the text. Additionally, there are extraction libraries for Presentation file format such as Microsoft PowerPoint Open XML Presentation file (PPTX) and Email file format such as email (EML).

Optical Character Recognition (OCR) tool is used to recognize text from scanned documents, PDF documents, and images. This tool was invented in the late 1920s by Austrian engineer Gustav Tauschek [23]. OCR Python libraries used to read and extract data from images are tesseract-OCR and EasyOCR. Tesseract-OCR is a google wrapped tool that was developed at Hewlett-Packard (HP) Laboratories in 1984 and 1999 for a PhD research project [2]. EasyOCR [24] is an open-source software that extracts text and data at the same time from images. EasyOCR performs faster on GPU and better with numbers compared to Tesseract-OCR which performs faster on CPU and better with alphabet recognition.

Speech Recognition [25] is a library that converts audio or words that have been read aloud into readable text with several engines, application programming interface (API), online and offline. Speech Recognition was developed in the 1950s and 1960s by Bell Laboratories which can only recognize digits. Later, in the 1980s it was able to recognize hundreds of words using a statistical method Hidden Markov Model (HMM) used to improve the accuracy of the text. Speech Recognition was last updated in late 2017 by Anthony Zhang under BSD License. Speech Recognition is fast and fairly accurate for those who speak fast and slow in writing.

B. Topic Modelling

Normally, when working with a small number of documents, it is much easier to read through and find the different topics presented within those documents. However, in the case of big data, that task becomes tedious. The need for more automated processes for the same task becomes apparent. As a tool, NLP has been used for language understanding [26] and at the document level, one of the most successful ways to accomplish that mundane and tedious manual task is called topic modelling. Topic modelling is the process of learning, recognizing, and extracting topics from a collection of documents. There are different ways to apply topic modelling algorithms to documents and we will discuss some of the different algorithms. When modelling topics on a collection of documents, two basic assumptions apply, namely, (a) each document contains a mixture of different topics, (b) and each topic contains a collection of words.

Albalawi *et al*. [9] have done a study reviewing articles relating to topic modelling that were published between 2015 and 2020. From this study we will focus on the methods explored namely Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) [10], Non-Negative Matrix Factorization (NMF) [11], Random Projection (RP) [12] and Principal Component Analysis (PCA) [13].

LSA is an unsupervised learning model used to extract relationships from documents and it acts as a dimensionality method to reduce the dimension of the huge corpus of text data. NMF is a matrix factorization method where we ensure that the elements of the factorized matrices are non-negative. This method can perform both dimension reduction and clustering simultaneously. In RP, a random matrix is used, and the original high-dimensional data is mapped onto a lower dimensional subspace with the reduced time cost. This method delivers sparse results because fundamental structure of the original data is ignored, and this often leads to high distortion. LDA is a probabilistic model, and it is the most popular method in real-life applications for extracting topics,

it provides more accurate results, and it can be trained online. It also addressed other models' limitations, like the latent semantic indexing and probabilistic latent semantic indexing. The LDA model will be used for topic modelling in this paper.

C. *Text Similarity*

Text similarity is a task of identify similar documents using identical contents to measure the similarity score. There are different types of methods to measure similarity score, namely, cosine similar, and Jaccard similarity.

Cosine similarity is a metric that models text document as a vector. The metric can be applied to any text in a form of sentence, paragraph, or a whole document [3]. It measures the similarity as the cosine of the angle between two vectors [4]. Cosine similarity can only be computed for two vectors that of the same size, the two vectors would be similar if they are close in terms of both magnitude and direction. Mathematically cosine similarity can be calculated using the dot product between the vectors and dividing it by the multiplication of the norms, it is defined as:

$$\cos\theta = \frac{A.B}{||A||||B||} \quad (1)$$

Jaccard similarity is a determination of the association between two texts [5]. It is a statistical measure of similarity between two sample sets, the two sets are defined as the intersection divided by their union. Mathematically it is defined by [6]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

Term Frequency – Inverse Document Frequency (TF-IDF) is an algorithm that determines the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus [7]. The calculation determines the relevance of a given word in a particular document. TF measures the frequency of a word in a document, IDF is the inverse of the document frequency. Mathematically it is defined as:

$$tfidf(t,d,D) = tf(t,d).idf(t,D) \quad (3)$$

Where D represent the corpus of the documents, t is the number of times a word occurs in a document, and d is document (set of words).

III. Methodology

This section discusses the architecture of the toolkit and the implementation. Firstly, we explain the methods used to extract the data, secondly, we explain the topic modelling pipeline followed by the text similarity pipeline. Figure 1 shows the architecture of the toolkit. The toolkit contains APIs or methods that can be used by external programs which can send data in a form of images, audios, and different types of documents whereby the toolkit is able to determine document type and extract the data from the document then send back the textual data back to the external program. For text analysis, the external program can either request topic modelling or type of text similarity using the APIs or methods, then the toolkit will be able to return the requested model back to the external program.

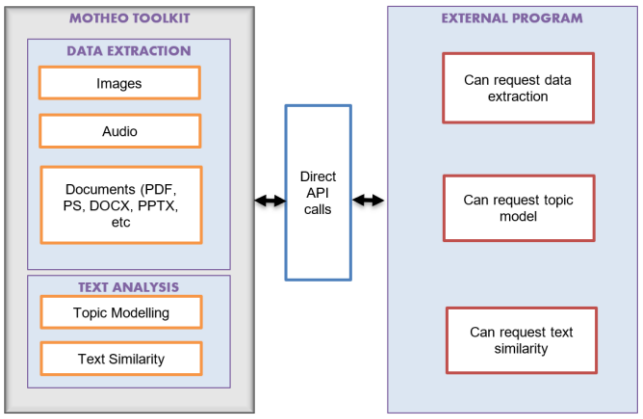

Figure 1: Overall Architecture

A. *Data extraction*

This section discusses the implementation of the data extraction methods on different types of files. These files are in a form of CSV, JSON, Image, XML, XLSX, HTML, XLS, DOC and DOCX, ODT and OTT, PDF, PS, TEXT, PPTX, RTF, EPUB, and EML.

- CSV/JSON/XML/XLSX/XLS: We use Pandas [1] library to process the CSV/JSON/XML/XLSX/XLS file and return the contents in a DataFrame[1] format.
- HTML: We use Beautiful Soup library to process the HTML file and extract the textual part excluding cascading style sheets (CSS) and JavaScript part.
- AUDIO: The toolkit support files in a form of wave or mp3. The toolkit converts the mp3 into wave form then we pass the audio to speech recognition engine. The toolkit supports the following speech recognition engines: Google Cloud Speech API, Microsoft Bing Voice Recognition, Wit.ai, Houndify, IBM Speech to text, and CMU Sphinx. We extract the textual part from the speech recognition engine. These engines need to be acquired before they can be used.
- Images: The toolkit supports wide range of various images listed in [8]. The toolkit can extract text from those images using a back-end software called tesseract [37].
- DOC/DOCX: The toolkit supports DOC and DOCX documents. It can extract the text from DOC using a back-end software called antiword [27] which must be installed on the system. For DOCX, the toolkit uses a software called python-docx [28] to extract the textual part.
- ODT and OTT: The toolkit supports both ODT and OTT documents. It can extract text from the documents using PYODF [21] library.
- PDF: The toolkit can extract text from PDF documents. A software called pdfminer [22] is used to process the PDF file.

[1] https://pandas.pydata.org/docs/user_guide/dsintro.html#dataframe

- PS: The toolkit can extract text from PS documents. We use a software called Ghostscript [29] to process the PS documents and then we extract the textual part.
- TEXT. The toolkit can read text files and return the text.
- PPTX: The toolkit can read Microsoft PowerPoint PPTX files. We use a software called python-pptx [30] to extract the text from each slide.
- RTF: The toolkit supports RFT documents. It uses unrtf [30] to extract the text.
- EPUB: The toolkit can read EPUB documents. It uses a software called ebooklib [31] to extract the contents which are in XML format. It then passes the contents to Beautiful Soup to extract the textual content.
- EML: The toolkit support EML documents. EML document contains plain text and HTML text. The toolkit extracts the contents of the EML file and pass the HTML part to Beautiful Soup to extract the text and combines with the plain text.

B. *Topic Modelling*

There are various algorithm exists to build topic modelling model. The toolkit uses Latent Dirichlet Allocation (LDA) algorithm. We tested the algorithm using a subset of the newsgroup dataset [33] which we used to train and test the model. The data was cleaned by removing stop words and special characters. The data was converted into bigrams and the words were lemmatized. The library uses Python library genism [34] to build a back-of-word model that is passed to LDA algorithm. We chose number of topics to be 20 when building the model.

C. *Text Similarity*

1) *Cosine similarity*

The toolkit contains methods that are used to read the input file, then pre-process the input file by removing non-alphanumeric characters, then converting the string to lower case, then apply lemmatization on each word. The methods also use TFIDF vectorizer model to convert input text into vectors. Lastly, the method passes the vectors into scikit-learn cosine similarity [35] function that returns a cosine similarity matrix computed across the whole input corpus.

2) *Jaccard similarity*

The toolkit contains methods to pre-process the input file by removing punctuation characters, and then converting the string to lower case, and then removing stop words, and then applying lemmatization on each word. The method applies Jaccard similarity across all the documents in the corpus and then returns a similarity matrix computed across the whole input corpus.

IV. How to Use The Toolkit

A. *Audio files*

As shown in Figure 2, the module must be imported on the environment, in this case speech module. The speech module accepts the following parameters:

- path: Is of type string, is a path of an audio file.
- mp3: Is of type Boolean, it indicates if an audio file is an mp3 file or not. By default, is set to nothing.
- key: Is of type string. The token can be acquired from Bing or Wit.ai. By default, is set to nothing.
- credentials_json: Is of type JSON. Is a file that contains google cloud credentials. By default, is set to nothing.
- client_id: Is of type string. Is acquired from Houndify. By default, is set to nothing.
- client_key: Is of type string. Is acquired from Houndify. By default, is set to nothing.
- username: Is of type string. Is acquired from IBM. By default, is set to nothing.
- password: Is of type string. Is acquired from IBM. By default, is set to nothing.
- Engine: Is of type string. Is used to specify speech recognition engine.

The module will return textual content in a form of a Python dictionary as shows in Figure 2.

```
>>> from motheo import speech
>>> speech(path="/home/ghost/Downloads/audio_files_harvard.wav", server="google")
{'alternative': [{'transcript': 'the stale smell of old beer drinkers it takes hea
 salt pickle taste fine with him tacos Al Pastore are my favourite is zestful fooc
  'the stale smell of old beer drinkers it takes hit to bring out the order a colc
 tacos Al Pastore are my favourite is zestful food is the hot cross bun'}, {'trans
 bring out the order a cold dip restores health and just a salt pickle tastes fine
 the hot cross bun'}, {'transcript': 'the stale smell of old beer drinkers it take
  a salt pickle taste fine with him tacos Al Pastore are my favourite exist for fc
  beer drinkers it takes hit to bring out the order a cold dip restores health anc
 favourite is zestful food is the hot cross bun'}], 'final': True}
```

Figure 2: Speech module

B. *Image files*

Figure 3 shows an image module imported into an environment. The image module accepts a file path of the image as input parameter. The module will return the text contents of the image.

```
>>> from motheo import image
>>> im = image("/image1.jpg")
>>> print(im)
YOU ARE THE
ARTIST OF
YOUR OWN

LIFE.
DON'T HAND
THE
PAINTBRUSH TO
ANYONE ELSE.
```

Figure 3: Image module

C. *Document files*

A document module is used to extract text from various types of documents. Figure 4 shows a document module imported into an environment. The module accepts the path of the file, and the sheet name (if document is Excel) as parameters. The module identifies the type of the file using file name extension.

```
>>> from motheo import document
>>> document("/sample4.xls", sheet_name='MyLinks')
   Unnamed: 0  ...                                        Unnamed: 2
0         NaN  ...                                               NaN
1         NaN  ...      Time-saving tools for pivot table power users
2         NaN  ...                Step by step instructions and videos
3         NaN  ...  Make instant backups, sort sheets, and many mo...
4         NaN  ...  Makes data entry easier when choosing from lon...
5         NaN  ...  Select single or multiple items from a listbox...
```

Figure 4: Document module

D. Topic modelling

The topic modelling module is used to create topic models for a given dataset. As shown in Figure 5, the module requires file path to data set and number of topics to create. The module may take long time to run based on the size of the data set. The module returns the topic model.

```
>>> from motheo import topicmodel
>>> t = topicmodel('/newsgroup.contents.truncated.txt')
>>> t.print_topics()
[(0, '0.121*"see" + 0.049*"seem" + 0.018*"wide" + 0.000*"stat" + 0
 0.000*"paragraph"'), (1, '0.077*"true" + 0.035*"fact" + 0.035*"se
 0.000*"sound" + 0.000*"re" + 0.000*"game"'), (2, '0.076*"still" +
 .000*"safety" + 0.000*"draw" + 0.000*"undoubtedly" + 0.000*"early
 0.001*"response" + 0.001*"proper" + 0.001*"disregardful" + 0.001*
```

Figure 5: Topic modelling module

E. Text similarity

The text similarity module contains cosine and Jaccard similarity techniques. The cosine similarity module shown in Figure 6 requires file path of the data set and returns the cosine similarity matrix computed across the whole data set.

```
>>> from motheo import cosinesimilarity
>>> cosinesimilarity('/similarity.txt')
array([[1.        , 0.38933573, 0.38933573, 0.15546222],
       [0.38933573, 1.        , 0.71143439, 0.        ],
       [0.38933573, 0.71143439, 1.        , 0.        ],
       [0.15546222, 0.        , 0.        , 1.        ]])
```

Figure 6: Cosine similarity module

The Jaccard similarity module shown in Figure 7 requires a file path of the data set to build the similarity index. Then model returns a Jaccard similarity matrix computed across the whole dataset.

```
>>> from motheo import jaccardsimilarity
>>> jaccardsimilarity('/similarity.txt')
array([[1.        , 0.375     , 0.375     , 0.1       ],
       [0.375     , 1.        , 0.71428571, 0.        ],
       [0.375     , 0.71428571, 1.        , 0.        ],
       [0.1       , 0.        , 0.        , 1.        ]])
```

Figure 7: Jaccard module

These modules can be used by any Python program that need to utilize the functions of the proposed toolkit.

V. Conclusion and Future Work

The proposed toolkit only support programs that are only on Python. Hence, for future work we recommend creating web application APIs that interact with internal functions. The web application API will be used as an interface for any software that needs to interact with the toolkit. The software does not have to be on Python, but it can be any software in any programming language residing at any location.

The limitations of some functionality of the toolkit is the dependency on Unix software platform which may not work on Microsoft Windows system. Hence, the future work will investigate and come with a solution for the tool to work on any operating system to make it independent.

References

[1] J. Reback, W. McKinney, J. Van Den Bossche, T. Augspurger, P. Cloud, S. Hawkins, A. Klein, M. Roeschke, J. Tratner, C. She and others, "pandas-dev/pandas: Pandas 1.4.1," Zenodo, 2022.

[2] R. Smith, "An overview of the Tesseract OCR engine," in *Ninth international conference on document analysis and recognition (ICDAR 2007)*, 2007, pp. 629-633.

[3] Rahutomo, Faisal, T. Kitasuka, and M. Aritsugi. "Semantic cosine similarity", in *The 7th international student conference on advanced science and technology (ICAST)*, vol. 4, no. 1, pp. 1, 2012.

[4] Xia, P., Zhang, L. and Li, F., 2015. "Learning similarity with cosine similarity ensemble," in *Information sciences* 307, 2015, pp. 39-52.

[5] Niwattanakul, S., Singthongchai, J., Naenudorn, E. and Wanapu, S. "Using of Jaccard coefficient for keywords similarity", in *Proceedings of the international multiconference of engineers and computer scientists*, 2013, pp. 380-384.

[6] J. Bank and B. Cole, "Calculating the Jaccard similarity coefficient with map reduce for entity pairs in Wikipedia", *Wikipedia Similarity Team*, vol 1, 2008, pp. 94.

[7] Ramos, J. "Using TF-IDF to determine word relevance in document queries", in *Proceedings of the first instructional conference on machine learning*, vol 242, 2003, pp. 29-48.

[8] A. Clark, 2022. [Online]. Available: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#fully-supported-formats

[9] R. Albalawi, T. H. Yeap and M. Benyoucef, "Using topic modeling methods for short-text data: A comparative analysis," *Frontiers in Artificial Intelligence*, vol. 3, 2020, pp. 42.

[10] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993-1022, 2003.

[11] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000, pp.556-562.

[12] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 245-250.

[13] H. Abdi and L. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433-459, 2010.

[14] N. Pradhan, M. Gyanchandani, en R. Wadhvani, "A Review on Text Similarity Technique used in IR and its Application", *International Journal of Computer Applications*, vol 120, no 9, 2015.

[15] T. K. Landauer, P. W. Foltz, και D. Laham, 'An introduction to latent semantic analysis', *Discourse processes*, vol. 25, no. 2–3, pp. 259–284, 1998.

[16] S. Gojare, R. Joshi, and D. Gaigaware, 'Analysis and design of selenium webdriver automation testing framework', *Procedia Computer Science*, vol. 50, pp. 341–346, 2015.

[17] D. Myers and J. W. McGuffee, 'Choosing scrapy', *Journal of Computing Sciences in Colleges*, v. 31, no. 1, pp. 83–89, 2015.

[18] V. G. Nair, Getting started with Beautiful Soup. Packt Publishing Ltd, 2014.

[19] "lxml: Processing XML and HTML with Python.", https://lxml.de/ (accessed Mar. 31, 2022)

[20] A. Shah "python-docx2txt.", https://github.com/ankushshah89/python-docx2txt/ (accessed Mar. 31, 2022)

[21] S. Roug "ODFPY.", https://github.com/eea/odfpy (accessed Mar. 31, 2022)

[22] Y. Shinyama “ODFPY.”, https://github.com/pdfminer/pdfminer.six (accessed Mar. 31, 2022)

[23] K. Hamad and M. Kaya , "A Detailed Analysis of Optical Character Recognition Technology", *International Journal of Applied Mathematics Electronics and Computers*, no. Special Issue-1, pp. 244-249, Dec. 2016

[24] "EasyOCR", https://github.com/JaidedAI/EasyOCR (accessed Mar. 31, 2022)

[25] A. Zhang, " Speech Recognition", https://github.com/Uberi/speech_recognition (accessed Mar. 31, 2022)

[26] M. Namazifar, A. Papangelis, G. Tur and D. Hakkani-Tür, "Language Model is all You Need: Natural Language Understanding as Question Answering," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7803-7807, doi: 10.1109/ICASSP39728.2021.9413810.

[27] A. van Os, " Antiword", http://www.winfield.demon.nl/ (accessed Mar. 31, 2022)

[28] "python-docx", https://github.com/python-openxml/python-docx (accessed Mar. 31, 2022)

[29] "Ghostscript", https://ghostscript.com/doc/current/Psfiles.htm (accessed Mar. 31, 2022)

[30] "python-pptx", https://pypi.org/project/python-pptx/ (accessed Mar. 31, 2022)

[31] "UNRTF", https://www.gnu.org/software/unrtf/ (accessed Mar. 31, 2022)

[32] "ebooklib", https://github.com/aerkalov/ebooklib (accessed Mar. 31, 2022)

[33] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, 1995, pp. 331-339.

[34] R. Řehůřek, P. Sojka, and Others, “Gensim-statistical semantics in Python”, Retrieved from genism.org, 2011.

[35] "Cosine similarity", https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html (accessed Mar. 31, 2022)

[36] "TextExtract", https://pypi.org/project/TextExtract/ (accessed Mar. 31, 2022)

[37] "Tesseract", https://github.com/tesseract-ocr/tesseract (accessed Mar. 31, 2022)