WIKIPEDIA WEB TABLE INTERPRETATION, KEYWORD-BASED SEARCH, AND RANKING

Kartikee Dabir
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Artificial Intelligence and Robotics Commons, and the Databases and Information Systems Commons

Recommended Citation
DOI: https://doi.org/10.31979/etd.ph9w-rpg9
https://scholarworks.sjsu.edu/etd_projects/1239

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.
WIKIPEDIA WEB TABLE INTERPRETATION, KEYWORD-BASED SEARCH, AND RANKING

A Project Report

Presented to

The Faculty of the Department of Computer Science
San Jose’ State University

In Partial Fulfillment

of the Requirements for the Degree Master of
Science

by

Kartikee Dabir

May 2023
The Designated Project Committee Approves the Project Titled

WIKIPEDIA WEB TABLE INTERPRETATION, KEYWORD-BASED SEARCH, AND RANKING

by

Kartikee Dabir

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE SAN JOSE STATE UNIVERSITY

May 2023

Dr. Ching-Seh Wu, Ph.D
Department of Computer Science

Dr. Chris Pollett, Ph.D
Department of Computer Science

Dr. Robert Chun, Ph.D
Department of Computer Science
ABSTRACT

**WIKIPEDIA WEB TABLE INTERPRETATION, KEYWORD-BASED SEARCH, AND RANKING**

by Kartikee Dabir

Information retrieval and data interpretation on the web, for the purpose of gaining knowledgeable insights, has been a widely researched topic from the onset of the world wide web or what is today popularly known as the internet. Web tables are structured tabular data present amidst unstructured, heterogenous data on the web. This makes web tables a rich source of information for a variety of tasks like data analysis, data interpretation, and information retrieval pertaining to extracting knowledge from information present on the web. Wikipedia tables which are a subset of web tables hold a huge amount of useful data, that if explored and mined appropriately, prove to be a rich source of important information for knowledge extraction. This research focuses on harnessing the capabilities of natural language processing for the task of information retrieval and data interpretation on the web, specifically web tables and more specifically, Wikipedia web tables to perform the tasks of keyword extraction, search, and ranking on them.

The goal of the project is to create an index using Wikipedia table data and title, to effectively search for pages that match input terms within a Wikipedia corpus and rank the results based on the frequency of presence of the term in the output. In nutshell, the system is a keyword-based search and ranking system. For this purpose, popular traditional NLP models RAKE, TextRank, TF-IDF, and SpaCy are utilized. The results are ranked from most to least relevant. Metrics including precision, recall, and F1 are further used to evaluate the performance of models.

A notable technological contribution made by this work is the creation of strategies for improving keyword extraction by leveraging the power of embedded web tables. According to the experimental findings it is observed that RAKE and TFIDF outperform TextRank and SpaCy in some cases and in other cases the opposite holds true. Furthermore, regardless of the model or dataset in consideration, on comparing the model performance scores with other experiments, our results outperform such studies. The results of this research work give insights into the efficacy of traditional NLP models in tabular text interpretation, keyword-based search, indexing, and ranking of relevant results. It is also found that the results of such an experiment vary largely depending on the dataset and the task at hand. Furthermore, the study also highlights the importance of selecting the right combination of models, dataset and tasks as well as the importance of correct preprocessing to get high performance.

**Keywords**— Keyword Extraction, Natural Language Processing, Information Retrieval, Similarity Score, Wikipedia Web Tables
ACKNOWLEDGMENTS

Working on this project, I acquired a lot of new skills thanks to the guidance of the excellent team of professors at SJSU. I would especially like to thank my project advisor Professor Mike Wu, who has been a great support in encouraging me to hone my skills throughout my master’s journey. His expertise in the field of Big Data Analytics and NLP and his invaluable inputs have proven to be a critical contribution to my work. He has been the guiding force behind this project, and it would not have been possible to put forth this work without his guidance. At every point in my academic and professional journey, I have needed a mentor who would guide me in the right direction and Dr. Wu has been that mentor in my master’s program.

I would like to take this opportunity to express my gratitude to Professor Chris Pollett and Professor Robert Chun for agreeing to participate in my work as committee members and providing their valuable inputs to improve this research. Their extensive knowledge, work, and vast experience in the field of computer science has made a lasting impact in my academic journey.

Finally, I would like to thank both my parents for being such pillar of support and encouraging me to face challenges head-on in this journey. They have helped me resolve every difficulty and supported me on good and bad days alike. Lastly a big thank you to my amazing brothers for being my strength and calm and to my loving husband for those words of encouragement and rock-solid support he has given me at every stage. It truly would not have been possible without their strong presence.
# TABLE OF CONTENTS

1 INTRODUCTION ........................................................................................................... 10

1.1 Overview .................................................................................................................. 13

1.2 Table Interpretation ................................................................................................. 13

1.3 Keyword Extraction for Table Interpretation ........................................................... 13

1.4 Problem Statement ................................................................................................. 13

1.5 Research Objective ................................................................................................. 13

1.6 Organization of Report ............................................................................................ 16

2 LITERATURE REVIEW .............................................................................................. 16

2.1 Web Tables and Information Retrieval ................................................................. 14

2.2 NLP and Wikipedia WebTables ............................................................................. 16

2.3 Preprocessing Task ............................................................................................... 17

2.4 TF-IDF ..................................................................................................................... 17

2.5 TextRank ................................................................................................................ 18

2.6 RAKE ....................................................................................................................... 18

2.7 SpaCy ..................................................................................................................... 19

2.8 Combination .......................................................................................................... 19

2.9 Evaluation Approaches ......................................................................................... 19

3 WIKIPEDIA WEB TABLES ...................................................................................... 21

4 TECHNICAL APPROACH ......................................................................................... 23

4.1 Indexing .................................................................................................................. 23

4.2 Three-Stage Process: Extraction, Search, and Ranking ........................................... 25

4.3 Python Modules ..................................................................................................... 31

4.4 Model Specific Modules ....................................................................................... 33

4.5 Development Platform ......................................................................................... 33

5 DATASET AND DATABASE .................................................................................... 35

5.1 Dataset ................................................................................................................... 35

5.2 Database ............................................................................................................... 36

6 THE FOUR NLP MODELS .................................................................................. 40

6.1 Overview ............................................................................................................... 40

6.2 Design .................................................................................................................... 41
LIST OF FIGURES

Fig. 1 Report Organization........................................................................................................... 12
Fig. 2 Summary of Literature Review.......................................................................................... 14
Fig. 3 Wikipedia Webtable Components ................................................................................... 21
Fig. 4 Index Creation in MongoDB ............................................................................................. 24
Fig. 5 Inverted Search Index in MongoDB Atlas ........................................................................ 24
Fig. 6 Auto Keyword-Based Indexing Example ......................................................................... 25
Fig. 7 The 3-Stage Process .......................................................................................................... 26
Fig. 8 Sample Wikipedia page in the Corpus ............................................................................. 27
Fig. 9 Wikipedia Web Table embedded within the page ............................................................ 27
Fig. 10 Mapping of the page and table in our dataframe .............................................................. 28
Fig. 11 Combined column Title – Content ............................................................................... 29
Fig. 12 Dataframe columns containing keywords .................................................................... 29
Fig. 13 Keyword-Based Search on the term 'Memphis' .............................................................. 30
Fig. 14 Index in MongoDB ......................................................................................................... 37
Fig. 15 Index creation code ........................................................................................................ 38
Fig 16 Flattened Data Mapping to MongoDB Database............................................................... 39
Fig. 17 Classification of the Models ........................................................................................... 40
Fig. 18 RAKE Search on the term 'Memphis' .......................................................................... 43
Fig. 19 Step-by-step RAKE Implementation ............................................................................. 44
Fig. 20 Undirected Graph of Terms .......................................................................................... 45
Fig. 21 Directed Graph of Terms ............................................................................................... 46
Fig 22 Step-by-step TextRank Implementation .......................................................................... 48
Fig. 23 Step-by-step TF-IDF Implementation ........................................................................... 50
Fig. 24 SpaCy pipeline overview .............................................................................................. 52
Fig. 25 SpaCy step-by-step implementation ............................................................................. 52
Fig. 26 SpaCy results .................................................................................................................. 53
Fig. 27 Average Precision .......................................................................................................... 55
Fig. 28 Average Recall .............................................................................................................. 57
Fig. 29 Average F1 Score .......................................................................................................... 59
LIST OF TABLES

Table 1 Components of WebTable .................................................................................................................. 22
Table 2 WebTable Classification ...................................................................................................................... 22
Table 3 Other popular corpus .......................................................................................................................... 36
Table 4 Key-Value Pairs in a single document in DB ...................................................................................... 36
Table 5 Precision Values .................................................................................................................................. 55
Table 6 Recall Values ..................................................................................................................................... 57
Table 7 F1 Score Values .................................................................................................................................... 58
Table 8 Summary of Scores .............................................................................................................................. 59
Table 9 Comparison with Related Work - 1 ........................................................................................................ 60
Table 10 Comparison with Related Work - 2 .................................................................................................... 60
Table 11 Comparison with Related Work – 3 .................................................................................................... 61
Table 12 Comparison with Related Work - 4 ..................................................................................................... 72
Table 13 Comparison with Related Work - 5 .................................................................................................... 72
Table 14 Comparison with Related Work - 6 .................................................................................................... 72
Table 15 Comparison with Related Work - 7 .................................................................................................... 72
1 INTRODUCTION

1.1 Overview

A web table, in its simplest form, is a table that exists on the internet. Web tables are embedded in web pages and differ from traditional tables in several ways. Firstly, they differ in their purpose. Traditional tables are created as efficient data storage platforms for fast querying and data analysis. Web tables, on the other hand, with the purpose of being a tool for ease of presentation and data management. Secondly, Web tables are heterogeneous in nature, organization, and structure when compared to the traditional relational table which have an underlying homogeneity. Lastly, web tables are embedded within webpages and traditional tables are usually a part of a database. Web tables are a common feature of most websites and are used to represent structured tabular data on the web.

1.2 Table Interpretation

Table interpretation is the study of methodologies that aim to make tabular data machine processable. Interpreting web tables involves analyzing the data presented and understanding its meaning and context. This process involves examining the table structure, understanding the variables and units of measurement, identifying patterns and trends in the data, and determining any relevant relationships between the variables. Effective interpretation of web tables can help individuals make informed decisions based on the presented data, such as in the case of comparing product prices or analyzing financial data. However, it's important to note that interpretation can be challenging, especially when tables are complex or the data is presented in a misleading way, making it important to approach web table interpretation with critical thinking and caution.

1.3 Keyword Extraction for Table Interpretation

Interpreting tables and extracting keywords are related in that both involve analyzing and understanding textual data. Keyword extraction identifies the most important words and phrases in a collection of related text. This process is useful for various tasks such as search engine optimization, text summarization, and content analysis. In the context of table interpretation, keyword extraction can be used to identify variables or column headings in a table and extract key information from table cells. By identifying the most relevant keywords in the table, individuals can see patterns and trends more easily in the data, which aids in the interpretation process. Keyword extraction is therefore a useful tool to aid in the interpretation of web tables and other structured data.
1.4 Problem Statement

Keyword-based search and ranking for interpreting web tables make use of relevant keywords to search web tables and the frequency of keywords to rank web tables based on their relevance to a specific query. This approach identifies the most important keywords or phrases in a collection of text and extracts them, uses these extracted keywords to index each entry in the table, matches the search term in a user query to these keywords, and returns the tables that match. Once relevant matching tables are identified, they are ranked based on factors such as keyword frequency, as in our work. This approach proves to be beneficial when searching for information in large databases or websites with many tables. By using keyword-based search and ranking techniques, individuals can more efficiently identify relevant pages and prioritize them based on their relevance to a particular query. In this way, efficient indexing on webtables is achieved with the use of contemporary NLP. The primary challenge is selecting the appropriate models, identifying relevant tasks, and harnessing the power of efficient preprocessing methodologies on the dataset at hand.

It is also important to note that keyword-based search, indexing, and ranking methods are not foolproof and additional analysis and interpretation may be required to fully understand and contextualize the information presented in the table. This study is majorly divided into three major tasks. The first major task is keyword extraction for keyword-based search task. The next task is the index-based search of entries of a table based on the extracted keywords. The final task involves ranking the returned matching table results based on the relevance i.e., how frequently they have the presence of the searched term in the query. Popular NLP models RAKE, TF IDF, TextRank, and SpaCy are used to perform keyword extraction, search, and ranking tasks on table data embedded within Wikipedia pages.

1.5 Research Objective

The goal of this study is to create a search and ranking system by keyword-based indexing of Wikipedia table data. Identifying the most relevant and essential phrases in each table textual entry is one of the fundamental goals for keyword extraction performed in this study. This can aid in summarizing the text's substance and providing a short overview of its primary themes and subjects. This in turn helps to develop a keyword-based search system that allows users to get relevant documents or web pages based on specified keywords or phrases. Furthermore, a rank of the relevant results is decided based on the frequency of the presence of search term in the indexed data. The motivation to use traditional NLP approaches such as RAKE, TextRank, SpaCy, and TF-IDF used in this study for keyword extraction and keyword-based search over AI ML-based approaches is that they are computationally efficient and simple in their application. They can swiftly detect key phrases and rate them
according to their importance in the text. Furthermore, these technologies are frequently language-independent, allowing them to be utilized for keyword extraction and search in a variety of languages. They can also handle a variety of text kinds, including news articles, social media postings, and scientific publications. The motivation behind using traditional NLP methods for keyword extraction and keyword-based search over AI ML-based approaches also comes from the extensive research work already present on these models.

1.6 Organization of Report

This report will be organized as follows. Chapter 2 of the review gives an overview of past and current work in the field of information retrieval, web tables, NLP in the field of IR, traditional NLP models, and keyword-based extraction. Chapter 3 explains the Wikipedia web tables in detail. Chapter 4 briefly about the technical approach used in the project. Chapter 5 deals with the dataset and database-related work in the project. Chapter 6 explores and explains the 4 traditional NLP models in depth and details their implementation. Finally, Chapter 7 gives a thorough overview of the experimental design and results. Chapter 8 concludes the research with a thorough explanation of future work that can be undertaken in this domain.
2 LITERATURE REVIEW

This section intends to study contemporary work around web tables and information retrieval on the web, by analyzing different literature based on the major tasks and models in our work. It also aims to study the work on evaluation parameters of these models while using them to perform the tasks. Using these approaches and their corresponding evaluation techniques, promising results have been achieved by different researchers. An incremental literature study has been built for the purpose of studying popular related work. The first subsection starts from the broad scope of analyzing literature related to web tables and information retrieval. The scope is narrowed down further in the next section, which talks about work specific to NLP, Wikipedia tables, and keyword-based search and ranking tasks. Further, the next section delves into work related to the different traditional NLP models for keyword extraction and keyword-based search namely RAKE, TextRank, TF-IDF, and SpaCy which have been implemented in this research. Next, studies related to the ranking task are analyzed using these models. Finally, in the last section, the current trends in the performance analysis of the models presented in different literature are studied.

The literature review is organized as follows: Section 1.1 dives deep into the research based on web tables and information retrieval; section 1.2 deals with literature pertaining to NLP and Wikipedia Web tables. Section 1.3 explains the preprocessing task done using NLTK. Section 1.4 is literature related to our first model TFIDF and its usage in keyword extraction, search, and ranking in literature. Section 1.5 deals with the TextRank model and Sections 1.6 and 1.7 explain the literature related to the RAKE and SpaCy. Section 1.8 gives us an overview of the literature related to the evaluation of results.

Figure 2 gives an overview of the organization of the literature review.
2.1 Web Tables and Information Retrieval

The world wide web consists of structured data in the form of HTML tables. The Webtables system by Cafarella et al. [1] introduced the idea of keyword-based search built on top of an existing search engine by proposing the idea of returning top k ranked results, using the web search engine and by extracting tables from those results that match the input query. Besides, the keyword search, the authors also introduce a model that records corpus-wide statistics on co-occurrences of schema elements and shows its use for joint graph traversal. At the core, the authors have devised a system that uses statistically trained classifiers to extract relational data from non-relational tables like HTML tables, which is a useful approach. Train and test data is generated to mark the HTML tables on the relational quality scale from 1 to 5 using two separate judges. A score of 4 means the table is highly relational. Embedded metadata is then extracted from these tables using a separate trained detector.
The metadata detector is fine-tuned to weigh precision and recall equally. Webtables relation search system makes use of features that are derived from the extracted relations and from the ACSDb, which is a statistical analyzer. For keyword ranking of extracted webtable relations, the authors use a series of ranking functions filterRank, featureRank, NaïveRank and coherency score. Each of the functions accepts the input query q and a parameter k which is the top k results. Each invokes the emit function to return a relation to the user. Zhang and Balog [2] address the information retrieval and more specifically the table retrieval problem by answering any information needed, by returning a ranked list of tables. For this purpose, the authors use two approaches based on the nature of the information needed: query-by-keyword and query-by-table. They pioneer the work of semantic webtable retrieval in this work, by returning keyword queries or returning existing tables against tables. The authors represent queries in dense and sparse vector representations and combine them with similarity measures. They use these as features in supervised machine learning. All the work is on done on Wikipedia tables. In another work, Zhang and Balog [3] put forth a survey that broadly encompasses the tasks of web table extraction, retrieval, and augmentation. The authors address 6 categories of information retrieval tasks: table interpretation, table extraction, table search, knowledge base augmentation, question answering, and table augmentation. The paper highlights the different components of information retrieval on the web. The search and table extraction tasks in this paper are crucial for this research. The paper states the feature-based keyword table search task, by detailing the query, table, and query-table features. Further, ranking using TF-IDF is elaborated. In the work [3], Wikipedia webtables are treated as a special case of web tables, and table extraction and search task are detailed. In this work, the different corpus of web tables and Wikipedia tables are listed. The authors explain the anatomy of a webtable, especially for Wikipedia tables in which the table elements present in a Wikipedia webtable are detailed. In yet another research, Zhang et al.[4] propose a new methodology for information retrieval from webtables using vector representations. They make use of webtable elements, like column headings, captions, and cells, to train entity or word embeddings. The embeddings are used in tasks like row and column population and table retrieval. These are added to existing retrieval models and are considered as additional signals to measure semantic similarity.
2.2 NLP and Wikipedia WebTables

For research work specific to Wikipedia tables, a closely related work is that of Bhagavatula et al. [6] who introduce WikiTables, an app that enables exploration of Wikipedia tables. For the keyword search on Wikipedia tables, the authors use a trained linear ranking model. It ranks these tables using four types of features: Page, Table, Table + Page and Query. In research by Muñoz et al. [7], the authors use an existing Linked Data knowledge base to find pre-existing relations between entities in Wikipedia tables and state that the approach extracts RDF triples from Wikipedia’s tables at high precision. The authors normalize Wikipedia tables, extract features, and map relations based on RDF triplets. In their work, DeMartini et al. [8] formulate a slightly different approach for Wikipedia table search, in which they base their search on entities that match types and attributes rather than traditional search. The authors use natural language processing to propose named entity recognition model to find entities in Wikipedia tables and develop an entity ranking system. Wikipedia tables are at the core of our research work. Understanding Wikipedia table structure, feature extraction and Wikipedia data mining from the above references is hence a crucial task. In their paper titled 'Using Linked Data to Mine RDF from Wikipedia's Tables' authors Muñoz et al. [7] present a method for extracting RDF triples from Wikipedia tables automatically. The method mines semantic information from Wikipedia tables and converts it into RDF triples by leveraging the structure and content of the tables. The approach initially finds the subject, predicate, and object columns in the table before employing a set of heuristics to extract the appropriate information. The retrieved data is then mapped to existing RDF vocabularies, with the resultant RDF triples being saved in a triple store. The authors demonstrate that their technique can extract correct RDF triples with excellent accuracy and recall on a dataset of over 200,000 Wikipedia tables. Authors Vu et al. [9] propose a graph-based approach for automatically inferring semantic descriptions of Wikipedia tables. The approach utilizes the structural and content-based characteristics of tables to construct a graph representation, where each node represents a column, and each edge represents a relationship between the columns. The authors then use graph-based clustering techniques to group columns that share similar semantic meanings and extract the semantic description of each cluster using a label propagation algorithm. The extracted semantic descriptions are then used to automatically annotate Wikipedia tables with rich semantic metadata. The authors demonstrate that their technique can infer accurate and relevant semantic descriptions with excellent precision and recall using a large dataset of Wikipedia tables. The study shows how graph-based strategies may be used to improve the semantic understanding and usability of Wikipedia tables. For NLP-specific work, the paper 'Understanding Tables in Context Using Standard NLP Toolkits' [10] is analyzed. The study provides a way for comprehending table context using natural language processing (NLP) techniques.
The method analyzes the text around a table using conventional NLP toolkits to discover key phrases and entities that are important to the table. The entities that have been detected are then utilized to enrich the table with semantic annotations such as entity kinds, relationships, and characteristics. The authors demonstrate that their technique may increase the accuracy of table categorization and entity recognition, as well as enabling more effective table retrieval and integration, using a dataset of scientific publications. The research shows how NLP approaches may be used to improve the readability and accessibility of tables in scientific publications.

2.3 Preprocessing Task

Loskutova [11] in work related to topic labeling, introduces the use of NLP NLTK to preprocess data, including tokenization, stop word removal, lemmatization, and stemming. The author further uses NLTK to achieve topic labeling and similarity score matching. NLTK is a popular python library for data preprocessing. Hardeniya et al. [13] and Perkins [14] in their respective books related to Python and NLTK, detail the usage of NLTK as an information retrieval model and as a data preprocessing model. The books list important data cleansing tasks like tokenization, stop-word removal, stemming, and POS tagging using Python and Python 3 for data preprocessing.

2.4 TF-IDF

TF-IDF is used for categorization and searches in this research. TF-IDF stands for. It is a weighted statistical model that gives an indication if a particular term is important to a document of a corpus. For example, the corpus can be a collection of Wikipedia articles, and the document can be an article. TF-IDF is quoted in most of the literature due to its scoring capabilities for the task of keyword extraction and keyword search. Vidal et al. [16] follow baseline traditional approaches to select keywords from web pages using Wikipedia as a source. They use TF-IDF and weighted words to extract keywords and return an ordered list of web pages. Biuk-Aghai and Ng’s work [17], which is critical in this study, is very similar to one of the approaches used in this work. Wikipedia articles are analyzed, and a set of keywords per Wikipedia category are extracted using a TF-IDF i.e., term frequency-inverse document frequency model proposed by them.

For the classification of an input document, TF-IDF weights are used to extract relevant keywords from the document, which are then matched to the keywords previously extracted from Wikipedia. The closest matching top-level categories are identified from all categories containing the document’s keywords. A cosine similarity metric is then applied to select the closest matching sub-category, recursing down the category hierarchy until the best matching categories are identified.
The result shows a set of categories that match the input document, along with a matching percentage value. These results are used to classify new documents that belong to a specific research area, or to classify a set of documents into topics and sub-topics, main keywords, as well as associated weights.

2.5 TextRank

The next popular model explored for keyword extraction and search, TextRank, was proposed by Mihalcea and Tarau [18]. TextRank, a graph-based algorithm can be successfully used in NLP tasks. Specifically, the authors propose two unsupervised methods for keyword and sentence extraction. The nodes are words, and the edges are relations between them. Keyword extraction using TextRank uses co-occurrence relations. According to Mihalcea and Tarau [18], two vertices are connected if their corresponding lexical units co-occur within a window of maximum words, N. TextRank was used by Li and Zhao [19] and is based on the PageRank algorithm. In this work, TextRank which is a graph-based algorithm, uses Wikipedia as a knowledge base to construct TextRank model to extract keywords. The authors Baruni and Sathiseelan [20] in their work show the usage of TextRank and a model called RAKE which has its base in NLTK for the task of keyword extraction and search. Rake refers to Rapid Automatic Keyphrase Extraction and is a popular traditional NLP model. It is an unsupervised, domain independent, and language-independent method for extracting keywords from individual documents and matching them by similarity score. A literature review on TextRank and RAKE and their importance in keyword-based search is undertaken by Yahya et al. [21] in which they state auto keyword extraction and analyze the performance and state the findings of these models.

2.6 RAKE

RAKE, which was invented by Rose et al. [22] for the purpose of auto keyword extraction, is stated in detail by the authors in their research work. RAKE begins keyword extraction on a document by parsing its text into a set of candidate keywords. After every candidate keyword is identified and the graph of word co-occurrences is complete, a score is calculated for each candidate keyword and defined as the sum of its member word scores. TextRank and RAKE both the models that were explored earlier, are popular for their keyword-based search capabilities. Also, a common observation in most of the papers referred to so far is that they use NLTK for their preprocessing requirements and use TF-IDF as a combination model. Authors Rinartha et. al [23] suggest two approaches for automatically extracting keywords from scientific publications in their work. The Rapid Automatic Keyword Extraction (RAKE) approach is a domain-independent method that extracts candidate keywords based on word co-occurrence in the text and then ranks the candidates based on frequency and degree of word-to-word connection. The Word Frequency (WF) technique, on the other hand, identifies possible
keywords using a domain-specific word list and then ranks them based on their frequency in the text. The authors assess the accuracy, recall, and F1-score of both algorithms against a dataset of scientific publications.

2.7 SpaCy

The SpaCy model by Porkodi [26] is used in NLP tasks and is used for keyword extraction using Python in this work. SpaCy converts data into objects and defines the pattern for relation extraction. It is also important to note that most of the work, ranking of relevant results is done based on their similarity matching score of the returned results. In their work, authors G, Suganya, and Porkodi R. [26] investigate the effectiveness of a machine learning model based on the Spacy library for extracting relations between items in the text. The model is trained on a dataset of annotated phrases, and its precision, recall, and F1 score, for relation extraction are evaluated. With an F1-score of more than 0.8, the findings indicate that the model can recognize and extract relationships between items with high accuracy. The authors also examine the model's performance for various sorts of relationships and propose opportunities for improvement. The study illustrates the use of machine learning-based techniques for relation extraction and emphasizes the need of assessing such models' performance in specific domains and scenarios. The findings can help to improve the accuracy and efficiency of relation extraction algorithms for natural language processing applications.

2.8 Combination

Some other works use a combination of two or more models or a completely novel approach. Rinartha et al. [23] and Lee et al. [24] use a combination of RAKE and word frequency in their work. Zuo et al. [28] utilize a combination of TextRank and Word2Vec. Azarafza et al. [31] and Yao et al. [32] use TextRank and TF-IDF combination for news-specific keyword search.

2.9 Evaluation Approaches

Like any research work, the evaluation of results and their performance metrics is an important parameter in determining the efficacy of the models. For keyword-based search and extraction different evaluation, metrics have been used by different authors. In their work, Khan et al. [29] use cosine similarity and similarity of all keywords extracted to compare the models. In another experiment, the authors assigned keywords and enhanced them using Wordnet. Moreover, they calculated the cosine similarity.
Authors Wu and Bhandary [30] in their work related to hate speech detection using ML, use Precision, Recall, and F1 scores of each of the models as evaluation parameters for their NLP task. A similar approach is used in our work to analyze results. In their work, Nguyen et al. [36] present a survey- evaluation of methodologies for the comparison of keyword extraction tasks with the use of datasets of different sizes and forms. Authors Ganiger and Rajashekharaiah[38], investigate the effectiveness of three common keyword extraction algorithms on a dataset of research articles: TextRank, TF-IDF, and RAKE. The authors examine each method's accuracy, recall, and F1-score, as well as their ability to extract meaningful and representative keywords from the articles.
3 WIKIPEDIA WEB TABLES

A preliminary understanding of the most crucial component that is at the heart of this work is necessary before diving into technical specifications. The type of tables analyzed in this research are Wikipedia web tables. These are special types of tables embedded within Wikipedia articles. A Wikipedia web table like any typical webtable is a collection of cells that have row and column structures. These tables are used to highlight important data and structure present in the page that is embedded. It is a tool for the ease of organization of structured data on a Wikipedia page. Another purpose it serves is that it is a form of visual communication for this page. Fig. 3 marks the different components of a Wikipedia webtable embedded within a page.

<table>
<thead>
<tr>
<th>List of Grand Slam men's singles champions</th>
</tr>
</thead>
</table>

Table 1 lists the different components of a typical Wikipedia webtable. It helps understand the anatomy of the table by highlighting a webtable’s different components.
Table 1 Webtable Component Details

<table>
<thead>
<tr>
<th>Sr.No.</th>
<th>Abbreviation</th>
<th>Component Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TH</td>
<td>Table Heading</td>
<td>Collection of labels that define what each column is represents</td>
</tr>
<tr>
<td>2</td>
<td>TC</td>
<td>Table Column</td>
<td>List of cells of a table lying vertically in the structure</td>
</tr>
<tr>
<td>3</td>
<td>TR</td>
<td>Table Row</td>
<td>List of cells of a table lying horizontally in the structure</td>
</tr>
<tr>
<td>4</td>
<td>TE</td>
<td>Table Entities</td>
<td>Tables contain entities, like as people, locations, and organization names. TE is a collection of all entities of the table</td>
</tr>
<tr>
<td>5</td>
<td>TP</td>
<td>Page Title</td>
<td>The table page title or the title of the web page that embeds the table</td>
</tr>
</tbody>
</table>

From the above-listed components, our research focuses on the table entities, which are the content of the table, and the page title. Each entity or cell is a separate row and is combined with the title of the page to form the data that extraction and search will be performed on. Cafarella et al. [1] provide the following classification of webtables, which classify the majority of the webtables present on the modern web today in Table 2.

Table 2 Webtable Classification [3]

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extremely small tables</td>
<td>Tables having fewer than two rows/columns</td>
</tr>
<tr>
<td>HTML forms</td>
<td>Used for aligning form fields for user input</td>
</tr>
<tr>
<td>Calendars</td>
<td>For rendering calendars</td>
</tr>
<tr>
<td>Non-relational tables</td>
<td>Low quality data tables like blank cells, simple lists</td>
</tr>
<tr>
<td>Relational tables</td>
<td>High-quality relational data</td>
</tr>
</tbody>
</table>

Wikipedia webtables are classified as relational tables. Many of the tables embedded within Wikipedia pages are relational in structure and have high-quality data. Hence, the data within them can be utilized to gain meaningful insights into what the table talks about. The task of preprocessing is undertaken to eliminate any empty cells and null data present within the tables. This leaves back high-quality relational data in the final dataset which is the ideal type of data used to gain meaningful insights.
4 TECHNICAL APPROACH

This section dives deep into the technical background of this research work. The two types of indexing undertaken on the data are thoroughly explained. The three important tasks of keyword/key phrase extraction, keyword-based search, and ranking of results based on the frequency of keywords are studied in detail. Next, popular python modules/libraries utilized throughout the work are explored and the development platform specifications are listed.

4.1 Indexing

Indexing is a crucial task in any information retrieval system. Indexing is the process of creating a catalog or a reference column of metadata, to the textual data or tabular textual data. This metadata is a summary of the most important topics highlighted in the collection of text. In its simplest form, an index in information retrieval is metadata related to a huge collection of text data. The purpose of indexing is the ease and speed of retrieval of matching data, from a huge data collection. During the task of indexing, the content of each table is analyzed to assign the relevant keywords or keyphrases that best describe the content of that table. The metadata i.e. the keywords contain an abstraction of the title and the tabular content present on that page. There are two types of indexing: manual indexing and automatic indexing. A human indexer assigns keywords or topic headers to a document during the manual indexing process. This procedure entails examining the document's content and picking relevant phrases that best represent the material. To maintain uniformity and correctness in the indexing process, the indexer may employ controlled vocabularies like as thesauri or topic heading lists. Automatic indexing, on the other hand, analyzes a document's content and assigns keywords or topic headers automatically. Natural language processing techniques are used by the algorithms to identify essential concepts, entities, or words significant to the content of the text. To increase indexing accuracy, automatic indexing may also include machine learning models that learn from previously indexed material. Indexing has been used in two places in this work. The first is while getting tabular data from the MongoDB database into the pandas dataframe, and the next is as a keyword index to the tabular data. Both the created indexes fall in the category of automatic indexing. The first index is a MongoDB inverted index on the ‘tableData’ field of the JSON Wikipedia data that is present in the database. It is called a search index in MongoDB. Embedded field titled ‘text’ present within ‘tableData’ field is accessed using this inverted index. This index is further accessed in python to create a column in the pandas dataframe. The table is indexed using the ‘db.collection.find()’ method that returns a cursor to the documents matching the query criteria, in this case the table text field of the data. Fig. 4 shows index creation in MongoDB.
Figure 5 illustrates the index creation process in MongoDB on the embedded field ‘text’ in the ‘tableData’ key-value pair.

The second index is essentially a smart index to the Wikipedia pages using the keywords extracted from the table content and title of the page. The combined column is what the extraction process is performed on and the newly formed column is now the index on which we will perform the search task. In this way, the keyword-based search is performed on the index containing only relevant keywords instead of the entire tabular textual data to
get the most relevant matches of Wikipedia pages. Such an indexed search is also faster than a regular search on the entire tabular data.

Proper usage of indexing can make a huge difference to the performance of an information retrieval system. Intelligent and proper indexing makes it easy to fetch relevant data fast. As highlighted previously, there are two types of indexing: manual indexing and automatic indexing. They differ in their process of categorization of data.

In manual indexing, the categorization is done by a human. Manual indexing is a time-consuming task but often produces accurate results. It is however not possible to categorize huge amounts of data manually. This is where automatic indexing helps. In automatic indexing a system or a machine with the use of software algorithms or artificial intelligence are responsible to put data in buckets. It is also known as machine indexing. With the advancements in software and AI, machines have become equally successful in giving accurate results to a task like categorization. Large amounts of data is indexed in a relatively short amount of time using automatic indexing. In our project, we are making use of automatic indexing in both the indexing tasks that we do. Figure 6 shows the automatic indexing we have done by extracting keywords from table data and title of the page.

![Fig. 6 Auto Keyword-Based Indexing Example](image)

### 4.2 Three-Stage Process: Extraction, Search, and Ranking

This research comprises of three major tasks which are implemented using each of the traditional NLP models selected as a part of this work the three major stages involved are:
1. **Keyword or Keyphrase extraction**

2. **Keyword-based search**

3. **Ranking based on keyword frequency**

A detailed explanation of each phase is provided in this section. Figure 7 lists these three stages in the order of their performance of these tasks:

![Fig. 7 The 3-Stage Process](image)

### 4.2.1 Keyword / Keyphrase extraction

The first crucial stage highlighted here is keyword/keyphrase extraction. This task deals with index creation on the table data. It is the stage in which the most significant relevant data is extracted from Wikipedia articles by analyzing the contents of the tabular data present on those pages combined with the page title. This extraction helps the system identify the main idea i.e., the topic that is highlighted in the Wikipedia page. This, in turn, helps in a way to index those pages. When a search is performed on this index, the relevant matching pages to the search term are identified. The four NLP models are put into action to accomplish this task. The models are TextRank, TFIDF, SpaCy and RAKE. Each of these models take a different approach towards keyword extraction. The upcoming section will specify the workings of each of these models in detail pertaining to the task of keyword extraction. Specific to Wikipedia tabular data, the process involves identifying the most relevant metadata present in the title of the page and content of the table and the creation of a catalog or index column after the extraction task. Figure 8 illustrates a sample Wikipedia page which is a part of the Wiki tables corpus.
Figure 9 shows a Wikipedia webtable embedded within the above page.
A combination of the page title and contents of the table extracted from the text field of the tableData column of the corpus is used for keyword extraction process. Figure 10 illustrates the mapping of the above listed page and table in the dataframe.

From Figure 10, an observation is that each row of the pandas dataframe is representative of one cell in the table. The title is repeated for each row in the dataframe since it is the same page that the tabular data belongs to. A combined column is created using table cell data and the title of the page. This combined column is the final column that keyword extraction is conducted on. Figure 11 shows the combination column mentioned above in the dataframe.
Further, the keyword-based search extraction task is performed. Figure 12 illustrates the data frame columns containing keywords for each of the NLP models, for the above Wikipedia table:
The specifics of how the above results are achieved using each of the NLP models are enlisted in the upcoming section.

4.2.2 **Keyword-Based Search**

The index column formed from the previous task of extraction is used in this particular phase to perform a search. An input term to be searched within the Wikipedia tabular data is selected. Next, the data frame rows are filtered out based on the keywords that match the search term. All relevant matching results are returned. The keyword-based search on Wikipedia data is an effective tool to gain insights from a huge collection of data present in Wikipedia. A table on a Wikipedia page generally summarizes the page content well and combined with the title of the page gives us crucial insights into the contents of the page. Figure 13 shows a keyword search on the term ‘Memphis’ and the results returned in our project.

![Fig. 13 Keyword-Based Search on the term 'Memphis'](image-url)
4.2.3 Ranking of results based on the frequency of keywords

The ranking of results stage is the third and final stage performed as a part of the three-stage process with the aim of returning relevant results ranked from most to least relevant. It is based on the frequency of the searched term in the extracted keywords. The previous stage of searching does not provide a sufficient measure to fulfill the purpose of this work since it is not sufficiently known which result is the closest match to the searched term. Hence, ranking of the results is undertaken using the same four models used in the process of extraction. In the ranking phase, the results are sorted based on the keywords extracted, in descending order of frequency of the keywords that match the searched term. This way a ranked list of results is returned, having the highest presence of the search term to the lowest presence. The task of ranking using specific models will be discussed in detail in the upcoming section.

4.3 Python Modules

Python is the language of choice for NLP model implementation and related tasks. This hold true for the purpose of this research. A large collection of NLP libraries is readily available in python, making it the obvious choice for NLP-related tasks. Moreover, data can be easily manipulated using python. Hence it is an ideal language choice to implement the tasks of keyword extraction and keyword-based ranking. This work uses the latest version of Python which is Python 3.0.

Following are the Python modules used in this project.

- **PyMongo**: The database is in MongoDB. Hence, the Pymongo library is used to make a connection between Python and MongoDB. MongoDB is a popular NoSQL database, and the PyMongo package allows developers to interact with it. It offers a simple and intuitive interface to perform CRUD - Create, Read, Update, and Delete operations on documents that are stored in MongoDB collections. In view of the flexibility and efficiency of its design, PyMongo is used in a wide range of applications including web development, data analysis as well as science. It is a flexible tool to work with complex data structures because of its support for different types of data, including binary records, regular expressions, and spatial data. PyMongo became an essential choice for Python developers who need to work with MongoDB databases because of its rich set of features, as well as the simple use of the API.
• **Mongo Client:** To connect to a MongoDB database instance and interact with its data, Mongo Client is a tool. It’s a driver that enables developers to use any programming language, like Python, Java, C++ and many others, to connect to MongoDB. The Mongo client contains a simple, efficient way to query and edit data stored in MongoDB collections as well as manage database connections and configuration.

• **Pandas:** Pandas is a Python Data Analysis and Manipulation library popular in the open-source community. For working with Structured Data, like spreadsheets, CSV files and SQL Databases, it enables the data structures to be used quickly, flexibly, or easily. We are using Pandas throughout the project for its dataframe creation and data manipulation capabilities. All the data for our project is held in the dataframes and we manipulate and store data in the dataframe columns. We create a separate data frame for each of the NLP models.

• **NumPy:** As a popular Python library for scientific computing, data analysis and machine learning, NumPy is very widely used. It gives a strong set of tools for manipulating large arrays and matrices which are often used in numerical calculations. The NumPy library is used for a variety of data preprocessing tasks throughout the project. Lambda functions are used for data filter and join tasks in our project.

• **NLTK for Data Preprocessing:** NLTK which is short for Natural Language Toolkit is a well-known library for NLP tasks in Python. It is open source and includes a wide range of tools to preprocess data, including tokenization, part-of-speech tagging (POS tagging), word propagation and lemmatization. In text mining, sentiment analysis and other applications that handle unprocessed textual data, NLTK has become an important tool. It has highly configurable algorithms, which allow developers to adjust them according to their particular use case. A corpus of text information, which developers can use to train models and test their algorithms, is also included in NLTK. The NLTK library is used for almost all the data preprocessing tasks in our project.
4.4 Model-Specific Modules

- **RAKE**: For the implementation of RAKE, majorly two libraries are used. The `get_ranked_phrases` and the `get_ranked_phrases_with_scores`.

- **Ipywidgets**: Throughout the project, the Ipywidgets library has been used for displaying a search bar and button for ease of visualization of results.

- **SpaCy**: For the implementation of SpaCy, the SpaCy library loads the en_core_web_sm for the implementation of keyword extraction. NLTK is used to get stop words and remove them from the data frame column.

4.5 Development Platform

**Local Desktop**: For initial storage of the JSON dataset file that we get via the download, the local storage on the MacBook hard drive is used. The JSON file is preprocessed to break it into chunks on the local platform, using Python script due to the large size of the WikiTables dataset. These chunks are then loaded one by one in the MongoDB database.

An overview of the local desktop configuration:

**Apple M1 Chip**

- 8-core CPU with 4 performance cores and 4 efficiency cores
- 7-core GPU
- 16-core Neural Engine
- 8GB unified memory
- 256GB SSD

**Jupyter Notebooks**: The environment used for development in this implementation is Jupyter Notebooks. Jupiter is a popular computational environment to implement a variety of machine learning libraries and tasks as well as NLP data manipulation tasks. A wide variety of functionalities like live coding, creating, sharing documents, visualizations, and text narration are present among its capabilities. Data science, scientific computing, and machine learning tasks use Jupyter Notebooks for analyzing data, and visualizing it to gain meaningful insights. Jupyter, which is a web-based interactive platform is an excellent choice for this study for its dynamic, feature rich capabilities. Enlisted are some of the technical specifications of Jupyter:
• **Memory/disk space:** 512MB RAM + 1GB disk space + .5 CPU core

• **Server overhead:** 2-4GB or 10% system overhead, 10GB disk space, .5 CPU cores

• **Port requirement:** port 8888
4 DATASET AND DATABASE

5.1 Dataset

The dataset selected for the purpose of this research is the WikiTables corpus, published by authors Bhagavatula et al. [6]. The WikiTables corpus is a dataset of approx. 2.4 million Wikipedia tables that are in JSON format. A collection of tables based on Wikipedia articles are included in the WikiTables dataset. Research related to Wikipedia tales using natural language processing, information retrieval and machine learning is facilitated by the availability of this large corpus. More than two million tables with related Wikipedia articles and a set of annotations are a part of the dataset.

The matrix of cells represents all tables in the dataset where each cell can have text, links, and images. The tables cover a broad range of topics, from sports statistics to scientific data as well as geographic information. A set of annotations for each table, including table headers, column and row headers, and cell types, such as text, number, or date, is also included in the dataset. For tasks such as table sorting and information extraction, this annotation makes it easier to use the database. The alignment of the Wikitables dataset with native text is a major feature. This means that each table is linked to a Wikipedia article, and the text of the article often provides context and additional information about the table. For tasks such as answering questions or creating text, this alignment makes it possible to use the data set.

For the purpose of creation of this dataset, authors Bhagavatula et al. [6] extracted HTML tables present on Wikipedia which had the class attribute labeled as “WikiTable” used to identify data tables. Using the Sweble parser, these tables were extracted from an XML dump of English Wikipedia articles published in November 2013. Sweble parser is a Java-based wiki markup parser library created by the Wikipedia Parsing team. It is intended to parse and analyze Wikipedia article markup in order to give a structured representation of the content.

The remaining 25 percent are used for training and validation purposes. The project corpus can be found at [6]. Table 3 shows a list of other popular corpora similar to the one we have used in our project (5th row). The other corpus is a collection of web tables, but our work is specific to Wikipedia tables which makes the WikiTables corpus a right fit for this study.
Table 3 Other popular corpus [3]

<table>
<thead>
<tr>
<th>Table corpora</th>
<th>Type</th>
<th>#Tables</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. WDC 2012 Web Table Corpus</td>
<td>Web tables</td>
<td>147M</td>
<td>Web crawl (Common Crawl)</td>
</tr>
<tr>
<td>2. WDC 2015 Web Table Corpus</td>
<td>Web tables</td>
<td>233M</td>
<td>Web crawl (Common Crawl)</td>
</tr>
<tr>
<td>3. Dresden Web Tables Corpus</td>
<td>Web tables</td>
<td>174M</td>
<td>Web crawl (Common Crawl)</td>
</tr>
<tr>
<td>4. WebTables</td>
<td>Web tables</td>
<td>154M</td>
<td>Web crawl (proprietary)</td>
</tr>
<tr>
<td>5. WikiTables</td>
<td>Wikipedia tables</td>
<td>1.6M</td>
<td>Wikipedia</td>
</tr>
<tr>
<td>6. TableArXiv</td>
<td>Scientific tables</td>
<td>0.34M</td>
<td>arxiv.org</td>
</tr>
</tbody>
</table>

5.2 Database

The dataset is loaded in a MongoDB database and an inverted index is implemented on it, to extract the embedded ‘tabledata’ key-value pair and the text column within it. This is the data on which the keyword extraction, keyword-based search, and ranking task is performed. Hence, embedded table data is one of the key-value pairs keyword extraction is performed on. The second key-value pair used is the title of the Wikipedia page. The extraction, search and ranking using all the models is done on a combined column of these two values from the corpus JSON which is stored as a key-value pair in the MongoDB database. The JSON file is loaded in a MongoDB Atlas database. A database named ‘KeywordSearch’ has been created for the purpose of storage. In the database, a collection called ‘MyData’ is used which contains the entries from the JSON file. Table 4 shows the key-value pairs in a single MongoDB document:

Table 4 Key-Value Pairs in a single document in DB

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Key-Value Pair</th>
<th>Description</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>_id</td>
<td>Default mongodb id of the document</td>
<td>Numeric String</td>
</tr>
<tr>
<td>2.</td>
<td>numCols</td>
<td>Number of columns in the table</td>
<td>Integer</td>
</tr>
<tr>
<td>3.</td>
<td>numDataRows</td>
<td>Number of data rows in the table</td>
<td>Integer</td>
</tr>
<tr>
<td>4.</td>
<td>numHeaderRows</td>
<td>Number of header rows in the table</td>
<td>Integer</td>
</tr>
<tr>
<td>5.</td>
<td>numericColumns</td>
<td>Number of columns in the table</td>
<td>Array</td>
</tr>
</tbody>
</table>


<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>order</td>
<td>chronological order of edits made to a particular article</td>
<td>Float</td>
</tr>
<tr>
<td>7.</td>
<td>pgId</td>
<td>Id of the page</td>
<td>Integer</td>
</tr>
<tr>
<td>8.</td>
<td>pgTitle</td>
<td>Title of the page</td>
<td>String</td>
</tr>
<tr>
<td>9.</td>
<td>sectionTitle</td>
<td>Title of a section within the page</td>
<td>String</td>
</tr>
<tr>
<td>10.</td>
<td>tableCaption</td>
<td>Caption of the embedded table</td>
<td>String</td>
</tr>
<tr>
<td>11.</td>
<td>tableData</td>
<td>Data present within table</td>
<td>Array</td>
</tr>
<tr>
<td>12.</td>
<td>tableHeaders</td>
<td>Header row of the table</td>
<td>Array</td>
</tr>
<tr>
<td>13.</td>
<td>tableId</td>
<td>Id of the table</td>
<td>Integer</td>
</tr>
</tbody>
</table>

‘TableData’ and ‘pgTitle’ are the key-value pairs of interest. To get the TableData in a dataframe an inverted search index is created on the MongoDB atlas database. Figure 14 shows an illustration of the index in MongoDB.

Fig. 14 Index in MongoDB
Figure 15 lists the code for the index creation in MongoDB Atlas:

```
1  {  
2     "mappings": {  
3         "dynamic": false,  
4         "fields": {  
5             "cTableData": {  
6                 "fields": {  
7                     "text": {  
9                         "type": "document"  
10                     }  
12                 }  
13             }  
14         }  
15      }  
16      },  
17      "storedSource": {  
18          "include": [  
19              "cTableData.text"  
20          ]  
21      }  
```

Fig. 15 Index creation code

The data from MongoDB flows into a data frame in the project environment. Further, flattening of the data frame is undertaken, to accommodate multiple entries of the tabular data. Each cell is now a row in the data frame. These rows are combined with the title, which is the final column for keyword extraction, search, and ranking process. Figure 16 lists an example of how the flattened data maps to our MongoDB database.
Fig 16 Flattened Data Mapping to MongoDB Database
6 THE FOUR NLP MODELS

6.1 Overview

Before diving into the theory and implementation of the four NLP models, it is important to understand the classification of these models and where our work fits into the bigger picture. This will help get an exact idea of the purpose of this study and the area of NLP that has been targeted. Figure 17 gives an overview of the classification of our four models and highlights the path of the models in this work:

![Classification of the Models](image)

Fig. 17 Classification of the Models

The classification begins with keyword extraction, which is a type of automatic keyword indexing. This is further categorized as supervised, semi-supervised, and unsupervised approaches. Traditional models fall in the category of unsupervised learning. TF-IDF and RAKE belong to the category of unsupervised methodologies utilizing simple statistics. TextRank is categorized as a graph-based approach while SpaCy is a combination of statistical and linguistic approaches. This work focuses on traditional models. The reason being, often with the surge in AI and ML, traditional models and their capabilities in doing tasks are neglected. Also, previous studies have used traditional models and our purpose is to study them on a novel dataset and compare results. Hence, unsupervised learning is used. That is further classified into simple statistical approaches, linguistic approaches, graph-based approaches and combination models that use more than one of these approaches. Next, a detailed explanation of work related to the four models RAKE, TextRank, TFIDF, and SpaCy is provided.
6.2 Design

The system will be designed as a Wikipedia search and rank system where the input query will be a keyword, or a set of keywords and the output will be a list of matching results based on relevance. The matching will be based on the extracted keywords and the frequency-based score of keywords. The results list will be a ranked list of Wikipedia articles, sorted by relevance and ordered by the most to least relevant results.

6.3 Installations

The work begins with the installation of three important Python libraries that have been used throughout the project. Pymongo, Pandas, and NumPy. Pymongo is used for the connection between the MongoDB database and Python, pandas to maintain the different dataframes throughout the project, and NumPy for the different preprocessing tasks done before keyword extraction.

6.4 Database Connectivity

MongoClient facilitates connection using the client string or MongoDB URI to the database titled Keyword Search.

The URI looks as follows:

```
MongoClient ('mongodb+srv://Kartikee:*Password*@keywordsearch.b0iwhey.mongodb.net/?
retryWrites=true&w=majority')
```

Next, the collection ‘myData’ is selected and indexed using the ‘find’ method of MongoDB, to load the table data in a dataframe column named ‘tableData’. Each data cell of the table is converted into a unique row of the dataframe for the convenience of execution of the upcoming keyword extraction process. The ‘explode’ method of the pandas dataframe is used to flatten the dataframe and create a new dataframe column. Null entries are further removed from the flattened dataframe. The intermediate dataframe is a column of dictionaries, which consists of cells in the tables. Next, each row is converted to a list since it is preferred to work with lists due to its features and ease of extraction process. NumPy is used to achieve this. Further, the tabular text column is converted to a column of strings. The title is combined with the table data for each row of the dataframe to form the final column. This column is accessed for keyword extraction and is titled ‘Title – Content’ which signifies the title of the page and contents of the Wikipedia table.
6.5 NLTK for data preprocessing

NLTK preprocessing capabilities have been exploited for a variety of tasks in this research. NLTK which is a crucial model for NLP tasks is used in various stages in each of the model implementations for tasks like tokenization, part-of-speech tagging, stemming, and lemmatization of the dataframe column i.e., the Title-Content column. It is the task of preparation of the data for the upcoming task of keyword extraction. NLTK is also used for cleaning and normalization of the dataframe column. Throughout the work, NLTK has been used for tasks like unwanted character removal, conversion of data to lowercase for matching, removal of null entries and empty strings in the data from dataframe and stop word removal for improving the accuracy of the keyword extraction process. Extracted keywords are also converted to strings for the purpose of compatibility, as input for certain models. The output of the NLTK data preprocessing task is hence, clean, and highly accurate data devoid of any errors.

6.6 RAKE

6.6.1 Theory

RAKE or Rapid Automatic Keyword Extraction model was first introduced by authors Rose et al. [22] in their research paper. RAKE can be used on a collection of text, for keyword extraction by detection and ranking of most important phrases and words. The core idea of RAKE is to detect possible keywords in the text by breaking them into singular phrases as well as terms. Next, RAKE scores them based on their match to the content. The importance of a phrase is determined by 2 key factors: the frequency it has and the extent to which it is related to other matching phrases in the document.

6.6.2 Implementation

Following are the detailed steps to implement RAKE (Rapid Automatic Keyword Extractor) for the task of keyword extraction, keyword-based search, and ranking:

**Step 1:** The input dataframe column content is preprocessed using NLTK by removal of punctuations and special characters.

**Step 2:** Next, the algorithm splits the list into keywords or phrases splitting at stopwords and delimiters and creates candidate phrases by using get_ranked_phrases method.

**Step 3:** Once the split on the text is complete a matrix of co-occurrences is built using this algorithm. Every row gives the number of times a term co-occurs with another term in candidate phrases.
Step 4: Each prospective keyword is granted a score, based on its frequency and position in the text and co-occurrence in the previously created matrix. The candidate keyword degree score is calculated by division of the total number of co-occurrences of the word and the frequency of the word. Calculate a combined score of each word for each candidate keyphrases. If two keywords or keyphrases appear together in the same order more than twice, a new keyphrase is created. The score of the key phrases is computed just like the one for a single key phrase.

Step 5: A keyword or keyphrase is kept so far as its score belongs to the top N scores. N is the number of keywords you want to extract. N defaults to 1/3rd of the content words in the document.

Step 6: Next, a particular term is searched within the keywords column to find all the matching rows and drop the duplicate matching results using the ‘drop_duplicates’ method of the data frame, since a flattened dataframe gives repeat results.

![Fig. 18 RAKE Search on the term ‘Memphis’](image)

Step 7: For visualization, the ipywidgets library is used that provides UI elements like a search box, buttons and results display area. An example of the search term ‘memphis’ within the dataframe is in the keywords column. The Title-Content column is returned as a part of the results in this example, but we also can return the entire dataframe matching rows.

Step 8: To rank results based on relevance for RAKE, the get_ranked_phrases_with_scores method is executed. This gives the scores associated with each term. A pair of scores and their corresponding keyword is generated in
a list. The dataframe is then filtered based on the search term and based on the second element of the tuple i.e the keyword. Once matching results are generated, the results are sorted in descending order of frequency based on the first element of the tuple, i.e., the score.

Figure 19 illustrates the step by step working of RAKE for our work.

Fig. 19 Step-by-step RAKE Implementation
6.7 TextRank

6.7.1 Theory

TextRank is a graph-based algorithm based on the principle of PageRank, which is another graph-based algorithm. Before diving into the detailed implementation, it is crucial to explore how the graph is used in the keyword extraction and ranking process. A graph has nodes and edges connecting these nodes. Typically, while representing text as a graph the terms are considered to be nodes of the graph and the edges are the relations between two words. These edges are directed edges for the task of representing word co-occurrence. The graph-based extraction of keywords is based on the idea of finding how crucial a particular word is i.e., the vertex in the graph is, based on the graph structure. This helps get the topmost k vertices ranked by importance. For word cooccurrence, the graph can be undirected such as in Figure 20.

The above graph gives a fair idea of cooccurring words but does not give a fair understanding of the in-degree (incoming edges of a graph) or the outdegree (outgoing edges of a graph), since it is undirected. Figure 21 shows a directed graph to solve this issue.
On graph construction, a decision is taken to choose the exact measures to keep the important vertices. The total degree of the vertex can be used as a measure, which is a combination of the in-degree plus outdegree divided by the max degree minus one. The formula to calculate the total degree is as follows:

\[
\text{Deg}(\text{total}) = \frac{\text{Deg}(\text{in}) + \text{Deg}(\text{out})}{N - 1}
\]

E.g., for the vertex having the word keyword, the degree is \(1 + \frac{2}{7} = 0.4286\).

Another measure that can be chosen is called neighborhood size. In this, a track is kept of the count of immediate neighbors of a node.

E.g., \(\text{NS chosen} = 3\)

Irrespective of the measure that’s chosen, the score of the vertex will determine its importance.
6.7.2 Implementation

The TextRank algorithm is implemented in this work, based on the above-mentioned graph-based ranking technique. The task at hand is of identifying the keywords from a dataframe column titled ‘Title-Content’. The following steps show the process to generate a list of keywords or phrases to identify the important topics highlighted in each Wikipedia table and rank them based on frequency using TextRank.

**Step 1:** Data preprocessing is undertaken in Step 1. The stop words, punctuations, any special characters and null entries are eliminated using NLTK.

**Step 2:** This cleaned data is tokenized into words or phrases. Tokenization is a text preprocessing technique in which a piece of text is broken down into individual units or tokens. These tokens are phrases or other meaningful terms present within the text. Tokenization is the first step in natural language processing tasks, as it helps to provide a standardized input for further analysis.

**Step 3:** A graph is constructed in which each token represents the node and the edges represent the co-occurrence of the token in a window of the text. Size is set on this window depending on text data length.

**Step 4:** In an edge connecting two terms, a weight gets assigned based on co-occurrence frequency. This weighted edge gives us an idea of the strength of the relationship within the two co-occurring terms.

**Step 5:** Each row and token in a row is looped over to score each token based on the weighted connection it possesses with the other terms. The degree of each word is calculated using the total of indegree and outdegree divided by the max degree minus one. This score is converged to get a final weight at every iteration.

**Step 6:** Like the previous model, top-ranking keywords are chosen to keep based on a number i.e., the number of top keywords to keep is pre-decided.

**Step 7:** Next, any keywords having no significant impact or a too long/short are filtered out to get the final output.

**Step 8:** For ranking using TextRank, a pair of scores based on frequency and their corresponding keyword is generated in a list. The dataframe is then filtered based on the search term and based on the second element of the tuple i.e the keyword. Once matching results are generated, the results are sorted in descending order of frequency based on the first element of the tuple, i.e., the score.
The following figure (Fig 22) details the step-by-step process of keyword extraction and ranking using TextRank.

![TextRank Implementation Diagram](image-url)

**Fig 22 Step-by-step TextRank Implementation**

6.7.3 Modules and their usage in implementation

TextRank implementation is initiated by installing Cmake and genism for the keyword module that is present in genism. Dataframe columns are split in tokens and using pytextrank which is a Python implementation of TextRank combined with the SpaCy pipeline the keyword extraction process is implemented. First, the en_core_web_sm module is loaded and the TextRank model to add to this NLP pipeline. Next, a method is defined to extract keywords to use this pipeline and extract the keywords from the Title-Content column and put them in a new column called keywords_TR. A search is then performed on this column after converting it to a string and the matching rows are returned. Like the last time, ipywidgets is used to display the results after the removal of duplicate rows.
6.8 TF-IDF

6.8.1 Theory

TF-IDF is a popular algorithm used to calculate the importance of a term in a collection of textual documents. It is known as term frequency-inverse document frequency. Term frequency represents the number of times a word appears in a text collection. This simply calculates the count. But certain words like ‘the’, ‘it’, and ‘and’ which can be termed as stopwords occur more frequently but are less important. Hence term frequency alone is not a sufficient measure. This measure is combined with inverse document frequency, to calculate the frequency of occurrence of a term i.e., how unique the term is in the word collection. Specific to keyword extraction the most relevant keywords can be identified using TFIDF. The score is calculated by multiplying the TF and the IDF. The top k high-scoring keywords are kept.

6.8.2 Implementation

The steps involved in TF-IDF calculation for the dataframe column containing ‘Title-Content’ are quite straightforward.

Step 1: Each row entry is broken into tokens or words. This process is commonly known as tokenization in NLP.

Step 2: For TF-IDF score calculation, the system iterates over each dataframe row for the particular page, and calculates the frequency of each of the tokens i.e. the term frequency.

Step 3: Calculating inverse document frequency involves calculating word importance. This is done by counting the rarity of occurrence of the token in the sentence in each row of the dataframe for a particular page.

Step 4: Next, the product of TF and IDF is taken to get the score. This score reflects how important the term is in context to the Wikipedia page that the table is embedded in.

Step 5: Keyword results are ranked based on their importance, using the score as a basis and the most important keywords are selected to be our final set of keywords.

Step 6: The dataframe is searched for the input term and the results are returned.

Step 7: Results are ranked by maintaining a tuple of the keyword and their corresponding scores. The first element is the keyword and the second is the score. Results are filtered on the first element and sorted descending based on the second element, which is the score.
The following figure (Fig 23) details the step-by-step process of keyword extraction and ranking using TF-IDF:

![Step-by-step TF-IDF Implementation Diagram](image)

**Fig. 23** Step-by-step TF-IDF Implementation

6.8.3 Modules and their usage for implementation

Sklearn’s feature extraction module is at the core of TF-IDF. The TfidfVectorizer of this module is imported for execution. This vectorizer is run on the columns, to remove stopwords. Next, the fit_transform method is executed to fit the Title-Content column. Feature extraction is done using the get_feature_name method. We get the feature index as follows:

\[
\text{feature_index} = \text{tfidf_matrix}[\text{doc_id}, :].\text{nonzero()}[1]
\]

The TF-IDF scores are calculated by zipping the feature index and the tfidf_matrix. These scores are sorted using a lambda function. Finally, the keywords are returned in the keywords_TFIDF column of the dataframe.
perform search task, the results that contain the search term are filtered out from keyword column and matching rows are returned. This visualization is done using ipywidgets library.

6.9 SpaCy

6.9.1 Theory

SpaCy is a popular model in NLP used for keyword extraction. With SpaCy, linguistic features of textual data such as POS (parts-of-speech), NE (named entity), and dependencies are used to identify the most important terms or phrases in a collection of textual data. The corresponding tasks are POS (parts-of-speech) tagging, NER-named entity recognition, and dependency parsing.

6.9.2 Implementation

The following steps are involved in the task of keyword extraction, search and ranking of results using SpaCy:

**Step 1:** Tokenization as previously discussed is the task of breaking the data into tokens or terms. Tokenization is applied to the dataframe column using NLTK.

**Step 2:** Each token is labeled with a POS tag i.e., which part of speech the word belongs to grammatically like noun or verb or adjective. It's an important task because some categories are more important in terms of the information, they give than others. For e.g., Nouns are more informative than pronouns.

**Step 3:** Dependency parsing identifies relationships between words belonging to a sentence. Each word is assigned a dependency label using ML in SpaCy. This tag is based on the syntactical relationship with other terms. This helps in recognizing important keywords.

**Step 4:** With NER, named entities like places, organizations, or people are identified in a collection of text. PERSON, ORG, or LOC are some of the named entities identified by SpaCy’s NER model and tagged. This process helps to get important keywords.

**Step 5:** Further, lemmatization is performed. It is the task of taking a word to its base form also called as lemma. This helps with the identification of different forms of the same word. All the words will be converted to its base form for ease of calculating frequency.

**Step 6:** Once all the above tasks are done, any of the above criteria or a combination of two or more of these criteria are used to extract the keywords and calculate word frequency. For example, we get all nouns and verbs related through a dependency relationship having the most frequency.
Step 7: The keywords are ranked to get the most important keywords for a particular row, and TF IDF is used to do so.

Step 8: Results returned are ranked based on the frequency of search terms in the extracted keyword column from highest to lowest frequency.

The following figure (Fig 24) lists the stages in a SpaCy pipeline.

![Spacy pipeline overview](image)

**Fig. 24 SpaCy pipeline overview**

Figure 25 illustrates the step-by-step process highlighted above.

![Spacy step-by-step implementation](image)

**Fig. 25 Spacy step-by-step implementation**
6.9.3 Modules and their usage for implementation

For the SpaCy model, the SpaCy library and its inbuilt en_core_web_sm SpaCy model are executed. This is the module that is at the core of this implementation. Next, dataframe column data is preprocessed by removing the stopwords and removal of special characters. This significantly reduces the amount of data that is fed to the NLP model, making it more efficient. The blanks are replaced with empty strings and duplicates are dropped before the actual model implementation. Finally, the SpaCy model is implemented using en_core_web_sm and by disabling the attribute ruler which is an element not used for fast processing speed of the model. A new column ‘spacy_tokens’ containing keywords is generated. Results are visualized using ipywidgets as in figure 26.

```
In [143]: import ipywidgets as widgets

# Define the function that will be called when the button is clicked
def search_term(sender):
    term = search_box.value
    response = task2.loc[task2['keywords_as_strings'].str.contains(term, case=False)].drop_duplicates()
    response_output.value = display(response['Title-Content'][0])

# Create the search box and button widgets
search_box = widgets.Text(description='Search Term: ', value='', layout=Layout(width='10cm'))
search_button = widgets.Button(description='Search')

# Create the output widget for the response
response_output = widgets.Output()

# Set the search button to call the search_term function when clicked
search_button.on_click(search_term)

# Display the widgets
display(search_box, search_button, response_output)
```

![Fig. 26 SpaCy results](image)

53
7 EXPERIMENTAL DESIGN AND RESULTS

7.1 Experiment #1: Precision

The accuracy of the extracted keywords is assessed using the precision score, a crucial assessment parameter for keyword extraction algorithms. It is determined by dividing the total number of extracted keywords by the proportion of successfully extracted keywords. In other words, it assesses the percentage of pertinent terms that the algorithm properly recognizes. An algorithm that extracts significant keywords with a high precision score does so with little inclusion of unimportant ones. In real life, obtaining high precision scores for keyword extraction is crucial for a variety of applications, including text categorization, summarization, and search engines. For instance, precision is crucial in search engines since consumers want the results to be accurate and relevant to their searches. Several non-relevant terms that are extracted by an algorithm might give inferior results for the search task and a negative user experience. Hence precision score continues to be a crucial component in the evaluation of the efficacy of keyword extraction algorithms. The precision score can be improved using different techniques. Using information that is specific to the domain in the algorithm, and using a lexicon that is a particular area such as an encyclopedia of law are some common techniques. Supervised machine learning can be used to label relevant and irrelevant data. All in all, a mixture of one or more techniques mentioned above helps improve the precision score. Average precision is calculated by using the ground truth keywords concept.

A list of relevant keywords that have been manually discovered and classified for a specific text or document is referred to as ground truth keywords. For this research, a model to count the frequency of each word in each row of the ‘Title-Content’ column is executed and the top n most frequent words as selected to be part of the ground truth keyword list. The average precision formula used to get the results listed in Table 5 is as follows:

\[
AP = \frac{1}{\text{count}_{\text{ground_truth}}} \times \sum_{i=1}^{\text{count}_{\text{extracted}}} (\text{precision}[i] \times \text{relevance}[i])
\]

where,

- \( \text{count}_{\text{ground_truth}} \) is the total count of relevant keywords in the ground truth
- \( \text{count}_{\text{extracted}} \) is the total count of keywords extracted by the algorithm
- \( \text{precision}[i] \) is the precision of the algorithm at the \( i \)th extracted keyword position
- \( \text{relevance}[i] \) is a binary value indicating whether the \( i \)th extracted keyword is relevant or not. It will be set to 1 if relevant, 0 if it is not.
The outcomes we get using average precision are listed in Table 5.

Table 5 Precision Values

<table>
<thead>
<tr>
<th></th>
<th>RAKE</th>
<th>TextRank</th>
<th>TFIDF</th>
<th>SpaCy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>96.17</td>
<td>40.03</td>
<td>94.26</td>
<td>87.23</td>
</tr>
</tbody>
</table>

Graph in Figure 27 shows the comparison of precision scores of the different models:

![Average Precision Graph](image)

Fig. 27 Average Precision

### 7.2 Experiment #2: Recall

Another crucial assessment parameter for keyword extraction algorithms is the recall score, which assesses how comprehensive the extracted terms are. It is described as the proportion of accurately extracted keywords to all relevant terms found in the corpus of text. In other words, it assesses the percentage of pertinent terms that the algorithm properly recognizes. A high recall score means that the algorithm is successful in locating all of the pertinent keywords while reducing the number of ones that are missed. High recall scores for keyword extraction are just as crucial in many applications as high precision scores. For instance, recall is essential in text categorization and summarizing since it makes sure the most crucial terms are not overlooked. The quality and accuracy of the classification or summary will suffer if an algorithm only retrieves a tiny portion of the pertinent keywords. The performance of keyword extraction algorithms is therefore significantly influenced by recall score.
The recall score of keyword extraction algorithms may be raised using a variety of methods. One strategy, for instance, is to integrate several keyword extraction approaches, such as rule-based, statistical, and linguistic methods. Another method is to leverage domain-specific expertise and heuristics to pinpoint the most crucial keywords, for example by leveraging a document's title, abstract, or metadata to direct the extraction process. Overall, depending on the application area and the unique needs of the task, attaining good recall scores for keyword extraction necessitates a careful balancing act between various strategies and approaches.

To compute the recall score based on ground truth keywords, we compare the terms generated by the algorithm to the appropriate ground truth keywords. The proportion of relevant terms in the ground truth accurately detected by the algorithm is measured by recall. The recall formula is:

\[
Recall = \frac{\text{Number of correctly identified relevant keywords}}{\text{Total number of relevant keywords in the ground truth}}
\]

Calculating mean recall with ground truth keywords entails comparing keywords retrieved by a keyword extraction algorithm to a list of relevant keywords selected and labeled manually from a dataframe column (i.e., the ground truth). The average recall is a measure of how well the algorithm does on average in detecting all relevant terms. The procedure followed to calculate average recall using the ground truth keyword technique is to compute the recall score for each row in the dataframe for the column containing keywords and average the values over all these columns. Following are the steps to calculate average recall:

- Calculate the frequency of each term in each row and select top k terms as ground truth keywords
- For every row containing keywords and ground truth keywords, compare the keywords determined by the algorithm to the ground truth keywords.
- Count the number of matching keywords from the ground truth that were successfully recognized for each row.
- Calculate the recall score for every row by applying the recall formula.
- For average recall calculation, add the recall scores from all columns of the dataframe.

Following is the formula for average recall:

\[
Average \ recall = \frac{1}{N} \cdot \sum_{i=1}^{N} (recall_i)
\]
where,
- \( N \) is the total number of documents or texts
- \( \text{recall}_i \) is the recall score for the \( i_{th} \) document or text

Average Recall Scores for each of the NLP models are listed in Table 6.

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAKE</td>
<td>76.97</td>
</tr>
<tr>
<td>TextRank</td>
<td>61.46</td>
</tr>
<tr>
<td>TFIDF</td>
<td>79.73</td>
</tr>
<tr>
<td>SpaCy</td>
<td>76.96</td>
</tr>
</tbody>
</table>

The graph in Figure 28 shows the comparison of the average recall scores of the different models.

7.3 Experiment #3: F1 Score

The F1 score, which combines the precision and recall scores into a single measure, is a frequently used assessment statistic for keyword extraction algorithms. It is described as the harmonic mean of recall and precision, which equally emphasizes each statistic. An increase in the F1 score, which goes from 0 to 1, indicates improved performance. An algorithm with a high F1 score will be successful in reducing irrelevant keywords and discovering useful ones. F1 score, which balances the trade-off between precision and recall, is a crucial indicator for assessing the overall performance of keyword extraction algorithms in practice. For instance, a high precision but poor recall algorithm can overlook crucial keywords, whereas a high recall but low precision algorithm might extract a lot of useless terms. Therefore, compared to accuracy or recall alone, F1 score offers a more thorough
assessment of the algorithm's performance. The F1 score of keyword extraction algorithms may be raised using a variety of methods. To increase both precision and recall, one popular strategy is to combine various keyword extraction methods, including statistical, linguistic, and rule-based approaches. Another method is to leverage domain-specific expertise and heuristics to pinpoint the most crucial keywords, for example by leveraging a document's title, abstract, or metadata to direct the extraction process. Overall, a high F1 score for keyword extraction necessitates the proper selection and fusion of many strategies and approaches, as well as a careful balancing act between precision and recall. To compute the F1 score for keyword extraction, a collection of ground truth keywords for a specific text or collection of texts is necessary. These keywords are created automatically using a function to calculate the frequency of each word in the dataframe column. Once the ground truth keywords are generated, they are compared to the model's extracted keywords. The precision is the fraction of correctly extracted keywords, whereas the recall is the proportion of correctly extracted keywords extracted by the model. The harmonic mean of accuracy and recall is then used to generate the F1 score as follows:

\[
F1 = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}}
\]

The average F1 score for keyword extraction is the sum of F1 values from numerous texts present within the dataframe column contents. This provides an overall performance metric of the model's ability to extract keywords properly. Table 7 lists the scores for average F1 scores for the models.

<table>
<thead>
<tr>
<th>Table 7 F1 Score Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAKE</td>
</tr>
<tr>
<td>85.5</td>
</tr>
</tbody>
</table>

Graph in Figure 29 shows the comparison of the average recall scores of the different models.
7.4 Comparison with related work

An important aspect of any experimentation is to validate the results found with other work that is similar to the work undertaken in the research. Table 8 summarizes the results of this research for ease of comparison:

<table>
<thead>
<tr>
<th></th>
<th>RAKE</th>
<th>TextRank</th>
<th>TF-IDF</th>
<th>SpaCy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>96.17</td>
<td>48.48</td>
<td>94.26</td>
<td>87.23</td>
</tr>
<tr>
<td>Recall</td>
<td>76.97</td>
<td>61.46</td>
<td>79.73</td>
<td>76.96</td>
</tr>
<tr>
<td>F1</td>
<td>85.5</td>
<td>48.48</td>
<td>86.38</td>
<td>81.77</td>
</tr>
</tbody>
</table>

In [19], the accuracy, recall, and F1 score for two algorithms, TextRank and TF-IDF, are reported based on their application on two datasets of short texts (news headlines and tweets).

Precision, recall, and F1 score for TextRank on the news headlines dataset were 0.80, 0.79, and 0.80, respectively. For the TF-IDF, the corresponding accuracy, recall, and F1 scores were 0.75, 0.72, and 0.74. The precision, recall, and F1 score for TextRank on the tweets dataset were 0.70, 0.69, and 0.69, respectively. For the TF-IDF, the corresponding accuracy, recall, and F1 scores were 0.66, 0.66, and 0.65. The study suggests a brand-new method for removing keywords from brief texts that makes use of Wikipedia as a knowledge base and the TextRank algorithm. It is important to note that this suggested approach TRW, outperformed TextRank and TF-IDF on both datasets in terms of accuracy, recall, and F1 scores. This is because it integrates Wikipedia relatedness with TextRank. The suggested approach obtained accuracy, recall, and F1 scores of 0.82 for the news headline dataset and 0.71 for the tweet dataset.
Table 9 compares our results from the results listed in the paper.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TR</td>
<td>TFIDF</td>
<td>TR</td>
</tr>
<tr>
<td>Precision</td>
<td>48.48</td>
<td>94.26</td>
<td>80</td>
</tr>
<tr>
<td>Recall</td>
<td>61.46</td>
<td>79.73</td>
<td>79</td>
</tr>
<tr>
<td>F1</td>
<td>48.48</td>
<td>86.38</td>
<td>80</td>
</tr>
</tbody>
</table>

If a comparison is done between the TextRank and TFIDF scores from this work with the author's work, we can see that TextRank's performance is equivalent to the author’s work and TFIDF performs better in our case, when we compare the scores. The major difference is the dataset that is being worked on, and the datasets chosen by the authors. Next, papers that use similar metrics as this work are analyzed for ease of comparison. The authors of a study [39] evaluated the effectiveness of each tool on a dataset of research papers. They discovered that TextRank beats TF-IDF and RAKE for keyword extraction in terms of accuracy, recall, and F1 score. The next study [40] compared the three algorithms on a dataset of news articles. In terms of accuracy, recall, and F1 score for keyword extraction, the authors discovered that TextRank and TF-IDF performed equally, whereas RAKE had lower precision but greater recall.

Table 10 compares the results of our work with [39].

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work[39]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td>Precision</td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td>Recall</td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td>F1</td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>

Table 11 compares the results of our work with [40].
Table 11 Comparison with Related Work – 3

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work [40]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>

Next, work that compares all four models are analyzed.

In the paper [22] the authors examined the performance of multiple keyword extraction algorithms, including RAKE, TF-IDF, TextRank, and a few more on a dataset of 40 papers from diverse domains. They discovered that RAKE and TF-IDF outperformed TextRank in terms of accuracy and recall of extracted keywords.

Table 12 compares results of our work with [22].

Table 12 Comparison with Related Work – 4

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work [22]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>

The authors of [41] compared the performance of multiple keyword extraction algorithms on a dataset of patent papers, including RAKE, TF-IDF, TextRank, and SpaCy. In terms of precision and recall of the retrieved keywords, they discovered that RAKE and TF-IDF beat TextRank and SpaCy.

Table 13 compares results of our work with [41].

Table 13 Comparison with Related Work – 5

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work [41]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>
The authors compared TextRank, TF-IDF, RAKE, and Spacy for automated keyword extraction in the study [42]. In terms of F1 score, they discovered that TextRank and SpaCy outperformed TF-IDF and RAKE. Table 14 compares results of our work with [42].

Table 14 Comparison with Related Work – 6

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work [42]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td>Precision</td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td>Recall</td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td>F1</td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>

Another study assessed the effectiveness of the four keyword extraction algorithms on a dataset of academic articles [43]. In terms of the F1 score, the authors discovered that TextRank and SpaCy performed better than RAKE and TF-IDF. Table 15 compares results of our work with [43].

Table 15 Comparison with Related Work – 7

<table>
<thead>
<tr>
<th></th>
<th>Our Work</th>
<th>Related Work [43]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAKE</td>
<td>TR</td>
</tr>
<tr>
<td>Precision</td>
<td>96.17</td>
<td>48.48</td>
</tr>
<tr>
<td>Recall</td>
<td>76.97</td>
<td>61.46</td>
</tr>
<tr>
<td>F1</td>
<td>85.5</td>
<td>48.48</td>
</tr>
</tbody>
</table>

It is observed that for all similar work from other authors, independent of the model or dataset under consideration when comparing model performance using precision, recall, and F1 scores with our trials, our results surpass the performance of all such research. RAKE, TFIDF, and SpaCy outperform by a significant amount and TextRank marginally outperforms in some cases and gives equivalent results in others. The findings of this study shed light on the usefulness of classical NLP models in tabular text interpretation, keyword-based search, indexing, and ranking of relevant results. This study concludes that the outcomes of such an experiment vary greatly depending on the dataset and the job at hand. It also emphasizes the significance of selecting the optimal combination of models, datasets, and tasks, as well as accurate preprocessing, to achieve high performance. This can be proven by the fact that when we check the results of the different studies above, TextRank and SpaCy outperform RAKE and TF-IDF and the opposite is true in some studies. Finally, in our study, the Wikipedia web tables dataset, the four NLP models used, and the tasks of keyword extraction, keyword-based indexing, and keyword-frequency-based ranking of results all complement each other to produce high-performance outcomes in our study.
8 CONCLUSION AND FUTURE WORK

In this project, we have proposed a novel approach to leverage the power of Wikipedia web table data to extract keywords from the data using four popular NLP algorithms namely SpaCy, Textrank, TF-IDF, and RAKE to perform keyword-based search and ranking of results based on the searched term. We used three evaluation parameters to evaluate the keyword extraction and ranking task needed for our search namely precision, recall, and F1 score. It is observed the RAKE model outperforms all the other models in the task with a high 96% precision, 86% F1, and 76% recall scores. This makes it an appropriate first choice of a model for our work. Close to follow is the TF–IDF model with a 94% precision, 86% F1, and 80% recall score. This means that the TF–IDF model performs almost equally well for our tasks. Next is the SpaCy model which performs slightly less than the RAKE and the TF-IDF models but better than the TextRank model. The Text Rank is the least performing model out of all the models at hand with low 48% precision, 48% F1, and 61% recall scores.

The performance does not match that of the other 3 models and has limitations in keyword extraction and ranking tasks. Hence in our work, the two models RAKE and TF – IDF are clear winners. In our work, the aim was to study the traditional NLP models for the task of keyword extraction. We wanted to study the effectiveness of these traditional algorithms for search and ranking using Wikipedia web table data.

A future enhancement can be to use neural networks models like YAKE and KeyBert for the task of keyword-based search and extraction. These model performances can be compared to the traditional model implementations to see which of the models is a more appropriate fit for the tasks of keyword-based search, extraction, and ranking. We can also widen the scope of our project by incorporating the different or all the elements of a Wikipedia page and then applying the models again on them. The same experiment can also be done on another corpus containing textual data and not restricted to tables. The reason we choose traditional NLP approaches over ML AI-based approaches is for decades, traditional natural language processing methodologies have been utilized to tackle a variety of language-related tasks such as sentiment analysis, information retrieval, and machine translation. To grasp the structure and meaning of human language, these approaches frequently depend on rule-based algorithms, statistical models, and linguistic understanding. Traditional NLP methods, unlike machine learning and artificial intelligence approaches, do not require massive volumes of annotated data to function well. They instead rely on expert-designed rules and models that have been developed over time to make them highly interpretable and explainable. Furthermore, classic NLP techniques are well-suited for jobs that need domain-specific expertise or language-specific subtleties, which ML and AI approaches may struggle to capture. Traditional NLP techniques, in general, continue to play an important role in language-related activities, particularly for applications requiring high accuracy, dependability, and interpretability.
LIST OF REFERENCES


