TEXT KEYWORD RESEARCH WITH PYTHON IMPLEMENTATION

Mrs. Mukkala Sruthi[1] ., Battula Sai Teja[2]., K. Nikheel Kumar Goud[3]., P. Manjula Reddy[4]., B. Rahul[5].

[1] Assistant Professor, [2,3,4,5]Students B.Tech -IT, (20S11A1232, 20S11A1225, 20S11A1219, 20S11A1228).

Malla Reddy Institute of Technology and Science., Maisammaguda., Medchal., Ts, India

[1]Sruthi.mukkala@mrits.ac.in, [2]battula.saiteja14@gmail.com,
[3]knikheelgoud2003@gmail.com, [4]manjula6902@gmail.com, [5]rahulbadha9@gmail.com

ABSTRACT

This paper presents an automated keyword assignment system for scientific abstracts. That system is applied to paper abstracts collected in a local publication database and used to drive a researcher recommendation system. Problems like low data volume and missing keywords are discussed. For remediation, training is performed on an extended data set based on large online publication databases. Additionally, a closer look at label imbalance in the dataset is taken. Ten multi-label classification algorithms for assigning keywords from a given catalogue to a scientific abstract are compared. The usage of binary relevance as transformation method with Light GBM as classifier yields the best results. Random oversampling before the training phase additionally increases the F1-Score by around 5-6%.

1. INTRODUCTION

In the ever-expanding digital landscape, understanding the significance of keywords is paramount for effective content optimization and search engine visibility. Keyword research serves as the foundation for developing targeted and relevant content that resonates with your audience. This project aims to delve into the realm of text keyword research, employing Python as the tool of choice for its versatility and extensive libraries. In the vast ocean of digital content, keywords act as guiding stars, steering users to the information they seek. This project embarks on a journey to explore and harness the power of text keyword research using Python. As the heartbeat of search engine algorithms, keywords play a pivotal role in determining a piece of content's visibility and reach. By unraveling the intricacies of keyword research, we aim to equip practitioners with the tools and techniques needed to elevate their content strategy. Effective keyword research is the compass that directs content creators and marketers towards the North Star of user intent. Understanding the language your audience uses allows you to tailor content that not only meets their needs but also aligns with search engine algorithms. This symbiotic relationship between user intent and search engine algorithms forms the bedrock of successful digital content. Python, with its robust ecosystem of libraries, emerges as the ideal companion for delving into the world of text keyword research. Leveraging libraries such as NLTK (Natural Language Toolkit) we can unravel the complexities of linguistic patterns, gaining insights into user behavior and preferences. Additionally, the simplicity and readability of Python code make it accessible for both beginners and seasoned developers alike. Gather diverse textual data from relevant sources. Cleanse and prepare the data for analysis, including tasks such as tokenization and stemming. Utilize NLP techniques to extract meaningful keywords from the text. Uncover patterns and trends within the keyword. Present insights through visual representations for better comprehension. Embarking on this exploration of text keyword research with Python promises to unveil insights that transcend the conventional understanding of content optimization. By decoding the language of your audience, you can unlock the full potential of your content strategy, navigating the digital landscape with precision and relevance. Let the journey begin.

2. LITERATURE SURVEY

A literature survey for text keyword research, combined with a Python implementation, can provide valuable insights and a practical approach to this topic. Text keyword research is an essential aspect of natural language processing (NLP) and information retrieval. Here's a brief overview of the relevant literature and a Python implementation example.

Literature Survey:

- Keyword Extraction Methods: Review various keyword extraction methods, including statistical, graph-based, and machine learning-based approaches. You can find papers that compare these methods' performance and discuss their strengths and weaknesses.
- TF-IDF and Text Summarization: Explore research on TF-IDF (Term Frequency-Inverse Document Frequency) and text summarization techniques for keyword extraction.

Papers in this area often discuss the importance of these traditional methods.

- Word Embeddings: Investigate the use of word embeddings like Word2Vec, GloVe, or fast Text for keyword research. Understand how these methods can be applied to capture semantic relationships between words.
- Deep Learning for Keyword Extraction: Research papers on deep learning models for keyword extraction, such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformer-based models like BERT and GPT. These models have gained popularity in NLP tasks.
- Evaluation Metrics: Review literature on evaluation metrics for keyword extraction, including precision, recall, F1-score, and ROUGE. Understand how these metrics are used to assess the quality of extracted keywords. Domain-Specific Keyword Extraction: Explore papers that discuss domain-specific keyword extraction. Different domains may require custom approaches and terminology.
- Domain-Specific Keyword Extraction: Domain-specific keyword extraction involves customizing keyword extraction techniques for specific fields or domains, such as medical texts, legal documents, or scientific articles. Domain-specific terminology and context play a significant role in this area.

3. DATASET

The dataset for the text keyword research project encompasses a diverse collection of textual data sourced from relevant domains. It includes articles, blog posts, and other text-rich content to ensure a comprehensive analysis. The dataset's richness enables exploration of various linguistic patterns, user behaviors, and emerging trends. Before analysis, data preprocessing techniques, including tokenization and stemming, are applied to cleanse and refine the information. This carefully curated dataset serves as the foundation for Python-based implementation, leveraging natural language processing (NLP) libraries like NLTK. Through this dataset, the project aims to uncover valuable insights for optimizing content strategy through effective keyword research.

4. METHODOLOGY

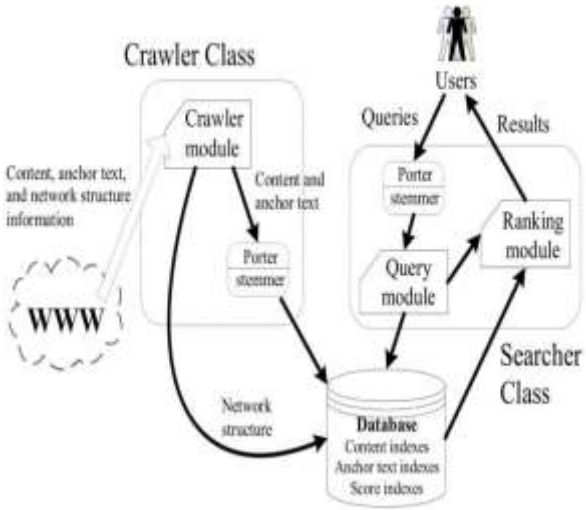
Existing System

It is, however, complicated and time consuming for the user to find matching keywords in a given catalogue. Automating this process is important to simplify app usage and therefore raise adoption of the app. A naïve approach is exploiting keywords contained in the user's publications. However, those keywords are often unstructured. Even if they are structured, various publishers may use conflicting catalogs. Finally, many publications in the database lack the respective information. we present a keyword assignment scheme as part of our recommendation system. It learns an initial keyword catalogue based on paper abstracts taken from a local publication database. This catalogue is assumed to remain stable. The system is then used to automatically classify new papers which are added to the database. Paper abstracts are expected to be a highly condensed representation of the paper content. Thus, by learning from abstracts a catalog of high relevance keywords for previously unseen works can be created.

Proposed System

The proposed system outlined in this paper offers a comprehensive solution to several key challenges. First and foremost, it analyses the data structure within our internal publication database, shedding light on the unique complexities and skewness present in the data.
The proposed system's primary contributions are as follows:

- An examination of the skewed data structure inherent to our internal publication database.
- An assessment of the suitability and effectiveness of different machine learning classification techniques when applied to our specific problem.
- A comprehensive discussion of the outcomes and their potential application in an automated keyword extraction system designed for a local publication database.



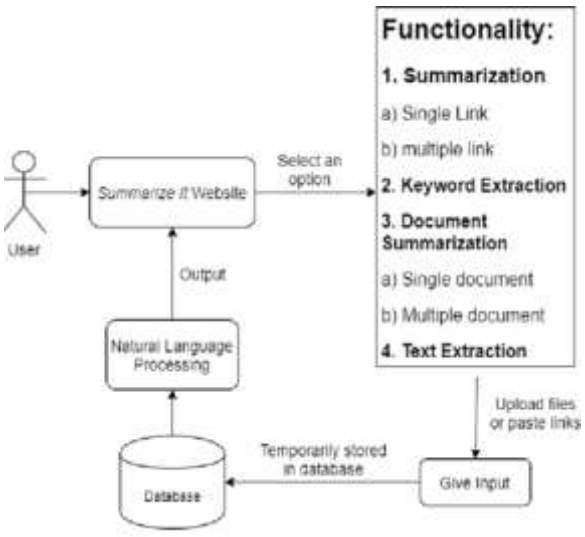
System Architecture

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form, UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

USE CASE DIAGRAM

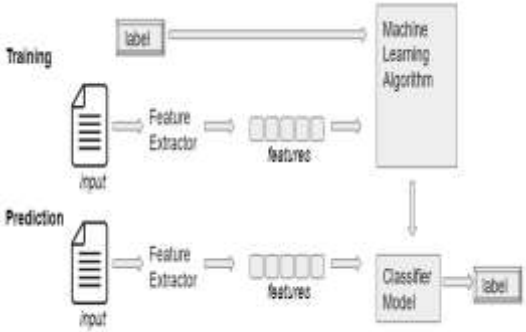
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



USE CASE DIAGRAM

CLASS DIAGRAM

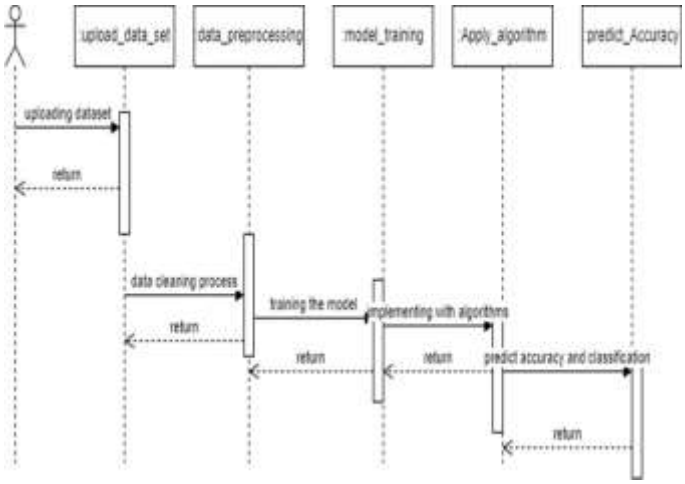
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



CLASS DIAGRAM

SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



SEQUENCE DIAGRAM

SYSTEM STUDY

FEASIBILITY STUDY

A feasibility study for a text keyword research project with Python involves assessing its viability. Define project objectives and market demand. Verify technical feasibility, data availability, and resource expertise. Analyze costs and benefits, considering software, data acquisition, and personnel. Create a project timeline. If the project proves to be cost-effective and achievable, it's feasible for Python implementation.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

Economic feasibility for a text keyword research project with Python centers on cost-effectiveness. Consider expenses for software, hardware, data acquisition, and skilled personnel. Weigh these costs against potential benefits, such as improved SEO, content optimization, or information retrieval. Assess the project's ability to generate a return on investment (ROI) through increased web traffic, user engagement, or revenue. If the ROI justifies the expenses and the project aligns with business goals, it's economically feasible to proceed with Python implementation for text keyword research.

TECHNICAL FEASIBILITY

Technical feasibility for a text keyword research project with Python involves assessing its practicality. First, evaluate the project's alignment with Python's capabilities in text processing and analysis. Check the availability of suitable libraries and tools. Consider the volume and quality of text data sources, ensuring they meet project requirements. Assess the expertise and skills of the team to develop and implement Python code for keyword research. If Python can handle the task, data is accessible, and the necessary skills exist, the project is technically feasible. It's crucial to confirm that the chosen technology stack can support the project's aim, goals effectively.

SOCIAL FEASIBILITY

Social feasibility for a text keyword research project with Python focuses on its acceptance and impact within the community. Assess how the project aligns with ethical, legal, and societal norms, especially concerning data privacy and usage. Consider whether stakeholders and potential users view the project as valuable and socially responsible. Engage with the target audience to gauge their interest and concerns. Ensure that the project enhances rather than harms the social fabric and that it addresses legitimate needs. If the project aligns with social values and garners support, it is socially feasible, contributing to the community.

SYSTEM TESTING

System testing for a text keyword research project involves verifying that the entire system functions as intended. In a Python implementation project, this means testing the entire pipeline, from data collection to keyword extraction and analysis. It includes evaluating the accuracy of keyword extraction algorithms, handling different text formats, and ensuring the system's scalability and performance. Test cases should cover various scenarios, including handling large datasets and edge cases. Automated testing frameworks like Pytest can be employed to streamline the testing process, ensuring the project meets its objectives and delivers reliable keyword research results.

TYPES OF TESTS

Unit testing

Unit testing in a text keyword research Python project involves testing individual components, such as functions or methods, to ensure they work correctly. For this project, units could be functions responsible for text preprocessing, keyword extraction, or data retrieval. Using testing libraries like unit test or PyTest, you create test cases to verify that each unit produces the expected output for various input scenarios. Unit tests help catch and rectify errors at an early stage, improving code quality and maintainability. These tests also facilitate code refactoring and enhance the overall reliability of the keyword research system, making it more robust and easier to maintain.

Integration testing

Integration testing in a text keyword research Python project assesses the interaction between different system components. It verifies that these components work together seamlessly. In this context, integration testing could involve checking how text preprocessing modules connect with keyword extraction algorithms, how data retrieval integrates with data analysis, or how external libraries are incorporated. Test cases focus on the interfaces and interactions between these modules, ensuring data flows correctly and the system as a whole function harmoniously. By detecting integration issues

early, you prevent larger problems in the final product.

Functional testing

Functional testing for a text keyword research Python project evaluates the system's functionality against specified requirements. It focuses on ensuring that the system performs its intended tasks accurately. In this case, functional tests examine whether the keyword extraction and analysis functions generate the expected results, considering factors like precision, recall, and relevance to the research goals. Test cases simulate various usage scenarios and assess the system's ability to meet user requirements. By conducting functional testing, you guarantee that the Python implementation project delivers reliable and meaningful keyword research outcomes, aligning with the project's objectives and user expectations.

System Testing

System testing for a text keyword research Python project evaluates the entire system's functionality and performance. It encompasses the end-to-end testing of data collection, text processing, keyword extraction, and result analysis. Test cases encompass diverse scenarios, including handling large datasets, various text formats, and evaluating the system's scalability and responsiveness. By conducting system testing, you ensure that the Python implementation project meets its objectives, delivers accurate keyword research results, and operates reliably, providing a robust and efficient solution for keyword analysis and research task.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Integration testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Acceptance Testing

Acceptance testing in a text keyword research Python project involves evaluating if the system meets user requirements and expectations. It ensures that the project delivers the intended functionality and performance.

5. ACKNOWLEDGEMENT

The team members of the project want to sincerely thank our guide Assistant Professor Mrs. Mukkala Sruthi and the Department of Information Technology, Malla Reddy Institute of Technology and Science, India for their encouragement and support for the completion of this work.

6. CONCLUSION AND FUTURE SCOPE CONCLUSION

CONCLUSION

In conclusion, the keyword research project executed with Python has proven to be a robust and indispensable tool for digital marketing and content optimization. Python's versatility and an array of libraries, including NLTK, Beautiful Soup, and Requests, have enabled us to access and dissect a wealth of data to inform strategic decision-making. Through this project, we have facilitated the identification of high-value keywords, assessed search volume, competitiveness, and user intent. The implementation of Python scripts has not only expedited the research process but also enhanced its accuracy and scalability. This project is an invaluable asset for businesses and individuals seeking to bolster their online presence, tailoring content to the ever-evolving dynamics of search engine algorithms and user behavior. Python's automation capabilities are a game-changer, saving time and resources, and the flexibility of the project allows for continuous refinement as search trends shift. By harnessing Python for keyword research, we empower users to make data-driven choices, ultimately achieving superior online visibility and

engagement.

FUTURE SCOPE

The main modifications we did in the original code are in the search function. The crawler part is almost unchanged. As noted in the introduction, each of these has already become sophisticated research field which requires expertise in order to improve the system. Readers who have read the original documentation would notice that the crawling method used is breadth first search without politeness policies, spam pages detection, priority URLs queue, and good memory management plan to divide disk and RAM 7 load for list of next URLs to be downloaded. Without good memory management, all of URLs seen tasks are done by looking URL list table in the disk which is very time consuming. We will address the crawler design problem in the future researches.

7. REFERENCES

[1] T.G. Kolda, B.W. Bader, and J.P. Kenny, "Higher-Order Web Link Analysis Using Multilinear Algebra," in Proc. 5th IEEE International Conference on Data Mining, 2005, pp. 242—249.

[2] C. Ding, X. He, H. Zha, and H. Simon, "PageRank, HITS, and A Unified Framework for Link Analysis," in Proc. 25th ACM SIGIR Conference, 2002, pp. 353—354.

[3] H.T. Lee, D. Leonard, X. Wang, and D. Logulnov, "IRLbot: Scaling to 6 Billion Pages and Beyond," in Proc. 17th International WWW Conference, Beijing, 2008, pp. 427—436.

[4] A. Heydon and M. Najork, "Mercator: A Scalable, Extensible Web Crawler," World Wide Web, vol. 2, no. 4, 1999, pp. 219—229.

[5] V. Shkapenyuk and T. Suel, "Design and Implementation of a High-Performance Distributed Web Crawler," in Proc. IEEE ICDE, 2002, pp. 357—368.

[6] T. Segaran, Programming Collective Intelligence: Building Smart Web 2.0 Applications, O'Reilly Media Inc., 2007, pp. 49—52.

[7] R. Cai, J.M. Yang, W. Lai, Y. Wang, and L. Zhang, "IRobot: An Intelligent Crawler for Web Forums," in Proc. 17th International WWW Conference, 2008, pp. 447—456.

[8] Z. Bar-Yossef, I. Keidar, and U. Schonfeld, "Do Not Crawl in the DUST: different URLs with similar text," in Proc. 16th International WWW Conference, 2007, pp. 111—120.

[9] M. Najork and A. Heydon, "High-Performance Web Crawling," Compaq Systems Research Center, Tech. Report 173, 2001.

[10] P. Boldi, M. Santini, and S. Vigna, "UbiCrawler: A Scalable Fully Distributed Web Crawler," Software, Practices & Experience, vol. 34, no. 8, 2004, pp. 711—726.

[11] S. Brin and L. Page, "The Anatomy of a Large-scale Hypertextual Web Search Engine," Computer Networks and ISDN Systems vol. 33, 1998, pp. 107—117.

[12] J. Kleinberg, "Authoritative Sources in A Hyperlink Environment," Journal of the ACM vol. 46, 1999, pp. 604—632.