Sage Plagarism Report

Powered by schoolworkspro.com

Krishna Prasad Adhikari 150265

15 Results Found

18.01 % Match Percentage

Submitted to: Softwarica	4.34%
Submitted to: Softwarica	4.19%
Source: C function Link: https://en.wikipedia.org/wiki/C_mathematical_functions	1.45%
Source: How to Use Naive Bayes for Text Classification in Python? Link: https://www.turing.com/kb/document-classification-using-naive-bayes	1.45%
Submitted to: Softwarica	1.19%
Submitted to: Softwarica	1.19%
Submitted to: Softwarica	0.78%
Source: What is data splitting and why is it important? Link: https://www.techtarget.com/searchenterpriseai/definition/data-splitting#:~:text=Data%20splitting%20is%20when%20data,creating%20models%20based%20on%20data	0.78%
Submitted to: Softwarica	0.62%
Submitted to: Softwarica	0.52%
Submitted to: Softwarica	0.41%
Submitted to: Softwarica	0.36%
Submitted to: Softwarica	0.36%

Submitted to: Softwarica 0.21%

Submitted to: Softwarica 0.16%

Link to github:
https://github.com/mikkelsinn/150265_KrishnaPrasadAdhikari_STW7071CEM

Table of Contents The Vertical Search

The Vertical Search Engine Crawler In the search engine, crawler script is written to crawl the seed URL provided in a systematic manner to discover and gather information from the web page (URL).

The crawler crawling through the web pages scans and extract all the contents.

The extracted contents are stored into database or file only after scrapping.

[CITATION Ols10 \l 1033] Scrapping focus on the specific set of data on the provided web pages.

From the provided URL " https://pureportal.coventry.ac.uk/en/organisations/school-of- computing-mathematics-and-data-sciences/persons/ " , only Publication Name, Publication URL, Author Name and Publish Dates will be selected.

Each of these information crawled, and scrap are stored in different JSON file system.

Crawler was executed on the provided URL and observed that there are 544 publications where Crawler has found 62 pureportal profiles.

All those crawled information are stored and the screenshot the files are attached below.

Figure 1.

Scrapper result after crawling a scrapping.

Crawler was executed in the profile section of School of Computing, Mathematics and Data Sciences.

The profile contains the make of publication, publication URL, Author name and publish date.

Those data can be observed in scraper_results.josn file and the sample content from the file is attached below:

Figure 2: Screenshot of sample JSON data

Figure 3.

Total publication crawled and time taken to crawl.

Crawler was executed manually, and data is stored in JSON file format.

Data stored in JSON file are unchangeable unless the contents in the webpage page are changed.
Unchanged data from the JSON file provide the same result for same query on every execution.
Explanation of crawler code:
Figure 4: Import package
In the above diagram, packages and modules required for crawling can be imported.
The “os” model is used to provide function for creating and removing directory, fetching the contents from the directory, identifying the current directory.
“Time” Module is used for representing time in code, “ujson” is used for json encoder and decoder.
The package “Beautifulsoup” is used for web scraping to pull the data out of HTML and XML files and “selenium”, “webdriver” is used for automating web browser interaction.
Python package “BeautifulSoup” is used for parsing HTML page.
The package creates a parse tree for parse pages which helps to extract data from HTML using web scraping.
Above attached script in screenshot scraps the contents crawled.
The result obtained from crawler will now be further processed by indexer.
Indexer In search engine, an indexer is responsible for creating index and facilitate the efficient information retrieval for searching.
[CITATION Wei11 \l 1033] Below mentioned steps are implemented as part of indexer.
i. Tokenization The result obtained from scraper contains the abstracted information.
The abstracted information should be further purchased to make the searching efficient and relevant.
So, the texts are breakdown into individual words or term called tokens and the tokens are further processed to convert them to their root form.
Figure 5.
Parsing and Normalization After tokenization, the words are further processed where special characters are removed.
Stops words are also removed and the words are converted into consistent case where upper-case letters are converted into lower case and normalization process is applied to ensure that individual words are treated as same form during search result processing.
Figure 6.
Removing Special Characters
In the above screenshot, special characters are defined and if there are any tokens containing those defined special character are omitted and only tokes are stored.
Figure 7.
Stemming Process
In the Stemming process, tokens containing upper case are converted to lower case.
Index Creation After tokenization, parsing and normalization of the text, indexes are created and frequency of text in document and location of text in document is also

added which will affect ranking of the search result.
Figure 8: Indexing
After indexing, different json file are created for every type of data.
Figure 9: Screenshot of indexed author dictionary
In the above screenshot of indexer result, “moham” is a term and numbers “{0,1,2,3,4,5,6,7,9,10,11}” is where the document is present.
Query Processor In this assignment, vertical search engine is developed.
So, in case of vertical search engine query processor is a key component for handling user queries and presenting meaningful search results.
In the vertical search engine, the search engine support key words and not Boolean queries are required to search.
Now, for searching, user is provided input form and users are prompt to insert the content they are searching.
[CITATION nyu23 \l 1033]
In search engine, certain constraints are applied where, user should provide least two words and a word must contain minimum of 3 characters.
For the query processer, the text provided by users should also follow processing steps where the query is first split into tokens.
Those tokens might contain special characters which are removed.
If there are any stop words, stops words are also removed and the words are converted into consistent case where upper-case letters are converted into lower case and normalization process is applied.
Also, the split texts are tokenized.
The screenshot of the process is attached below:
Figure 10.
Converting User Input to lower case
Figure 11: Screenshot of code where user input is taken and tokenized.
In the above attached screenshot, user input text (query) is converted to lowercase and spilt.
The user data is validated for at least two words and word should have at least 3 character and the words are tokenized using function word_tokenize.
Now for further improvement in searching of more appropriate result, NLTK module was imported, and stemming is used for processing morphological variant of root word.
The screenshot for which is attached below.
Figure 12.
Stemming of user input query
After stemming, the processed user input query is executed to search the result from the database.
Contents from the data file is extracted for presenting to the
Figure 13.
Query to fetch result from data file.
In the above screenshot, result is derived and sore in temporary file for presentation.
To make the result more relevant to the user query, ranking of the search result based on the searched terms found on the document is performed.

For the ranking purpose, cosine similarity is used.
Here, cosine similarities between two non-zero vectors are calculated.
Below mentioned formula is used to compute the cosine similarities.
Here A and B are two vectors:
Where A.B are product of A and B which is computed as sum of element wise product of A and B.
And ||A|| is L2-normalized data A and is computed as square root of the sum of squares of elements of the vector A and similar is the ||B||.
[CITATION Han12 \l 1033]
However, in python the cosine similarities in python is calculated using sklearn.metrics.pairwise library where contain cosine_similarity.
Screenshot of cosine similarities are attached below.
Figure 14.
Screenshot of importing library for cosine similarities
Figure 15.
Screenshot of code using cosine_similarities for ranking the result.
After fetching and ranking the search result, the result is ready to present to the user.
The obtained result is printed based on rank obtained.
Screenshot of the obtained result is attached below.
Figure 16.
Screenshot of search result
In the above attached screenshot result, author “Ian Dunwell” was searched and all the publications by “Ian Dunwell” are listed along with ranking of the result.
In the above screenshot, “Machine Learning” text is provided as an input in the search field.
Total of 71 results are provided by the search engine and total time taken to execute the query is 0.04s.
Also, we can observe that result with highest rank is listed on top.
Rank, Title, URL, Date and Author was printed for all the search result.
Also, we have “AND” and “OR” search rule where on the search engine, where, AND must satisfy that the all the query must match in the same document whereas OR rule print result for any match case.
To illustrate OR and AND rule, we try to search one word from one title and next word from different title.
In the above result, Tile on the first result contains motor and ranked 0.41 and is on top.
And next searched word multimodal in on other search results ranked based on calculated rank.
Now applying AND rule for the same input.
No result was found with AND rule as those two words does not belong to the page.
Subject classification (text classification) Estimating various parameters and comparing the performance of various models are two uses for training sets.
After the training is finished, the testing data set is used.
To ensure that the final model functions properly, the training and test data are compared.

Training data was collected manually from internet in CSV format.
The training data contains paragraph from different subject modules like Computer Science, Mathematics, Physics, Statistics, Data Structures and Algorithms, Machine Learning etc.
For the classification, Naïve Bayes Classifier Algorithm is used as Naïve Bayes Classifier being simplest and most effective classification algorithm in building quick machine learning models and prediction.
[CITATION IBM21 \l 1033]
To measure the algorithm performance, confusion matrix was used and the model was evaluated using standards of recall, precision, accuracy and F1 Score.
Pandas library is used to read the train and test datasets.
Now, from the train dataset, stop words are removed and stemming was performed, and the tokenization was performed.
Figure 17.
Code to remove stop words and stemming and tokenize.
Special character and stop words from the train data was removed.
And the data are categorized into computer science, physics, mathematics, and statistics.
From the available train and test sample, the selected categorized data are selected.
The length of input train and test data and output training samples was calculated.
Figure 18.
Input training and testing sample and output training and testing samples count
Now, prediction was performed to predict the accuracy which is a measure that provides percentage of correctly recognized classes.
The above-mentioned code predicts the accuracy and F1 score.
The result after accuracy and f1 Score calculation is attached below.
Here, 63% accuracy was calculated and F1 score was 76%.
Figure 19.
Heat map
Conclusion Hence, a vertical search engine was created.
The search engine crawl the pure portal of Coventry university and scrapping was done to collect publication name, url, author name and date.
After crawling, 64 profiles and 544 publications were found.
The crawled data perform Tokenization, removing stop words, removing special character, and stemming before indexing.
Indexing is performed on the processed data where indexes are created and frequency and location of text in document is added.
After indexing, query processing is performed to user query and extract result on the base of calculated ranking.
Text classification was also performed.
Train and test dataset was collected, and the dataset was also preprocessed to remove special character, stop words, stemming and was tokenized.
To solve the classification problem Naïve Bayes algorithm was used and prediction was calculated.
References Han, J., Pei, J., & Kember, M. (2012).

The Morgan Kaufmann Series in Data Management Systems.
In Data Mining (Third Edition) (pp. 39-38).
(2021, 03 01).
Naive Bayes classification.
Retrieved from https://www.ibm.com/docs/en/db2/9.7?topic=classification-naive-bayes
nyu.edu.
Efficient Query Processing in Large Search Engines.
engineering.nyu.ed: http://engineering.nyu.edu/~suel/queryproc/
Olston, C., & Najork, M. (2010).
Foundations and Trend in Information Retrieval.
WebCrawling, pp. 176-232.
Weideman, M. (2011).
A comparative analysis of search engine indexing time.
Peninsula University of Technology (pp. 7-10).
Cape Town: Cape Peninsula University of Technology.
Crawler.py
Indexer.py
QP_GUI.py
Classifier.py