International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I2P221

Volume 69 Issue 2, 144-159, February 2021
© 2021 Seventh Sense Research Group®

A Real And Accurate Semantic Search Indexing Approach Using Asvm Machine In Big Data Analytics

[1]Y.Krishna Bhargavi, [2]Dr. Yelisetty Ssr Murthy, [3]Dr.O.Srinivasa Rao,

[1]Gokaraju Rangaraju Institute of Engineering and Technology, Computer Science and Engineering, Hyderabad.
[2]SRKR Engineering College, Information Technology, Bhimavaram.
[3]University College of Engineering, JNTU Kakinada, Computer Science and Engineering, Kakinada.

[1]kittu.bhargavi@gmail.com,[2]yssrmurthy@gmail.com,[3]osr_phd@yahoo.com

Abstract

Big data is a receiver of information to give accurate search and relevant content for better work efficiency-experience. In earlier days, many semantic algorithms have been designed to improve effective content searching, but these are facing limitations. The information being retrieved and content filtering help future applications to get comfortable with operations. Web browsing and its recommendation systems are currently facing inaccurate content tracking, and hence the users cannot acquire the required information. In this research work, an adaptive SVM-based semantic search technique has been designed for big data applications. The method is calculating the performance measures like query time, building time, accuracy, average precision, stdError, SSR. Here, the presented KVASIR-ASVM architectural design encounters the existing systems and finally enhancing the accuracy to 99.72% and recalling at a rate of 0.997%. These experimental results outperform the methodology and compete with current technology.

Keywords: *semantic search, ASVM, bigdata, Internet, query time.*

I. Introduction

The semantic search technique is improving the traffic of websites and is simplifying the search engine functionality with relevant information. Search engines like Google, Yahoo, Opera Mini, Firefox, and Microsoft Edge are facing many ambiguities. The current technology is majorly working on the Internet and big data platforms. Bigdata offers much excessive information for advanced future applications. The smart content recommendation system is necessary to extract the needful articles from the web(Yahoo, Facebook, and BBC News). Many typical applications concentrate on semantic article search techniques for avoiding complex content filtering. From 2011, Google and other browsers started to move towards artificial intelligence and machine learning techniques for understanding the query and response from big data. A big data drive consists of a smart file handling system mechanism like Hadoop. In this, small and big sizes of the content can be differentiated by entity types. The semantic search continuously analyses the web pages and creates data

Relevant to entity type and properties. Using this concept, files can be easily handled in the big data platform. In this investigation, web and serves semantic search has been examined based on machine learning algorithms for big data analytics. Various search engines have many background links and include keywords. Because of this relevant keyword information, users do not attain the exact search keyword. The current technology is going rapidly, but today search engines are unable to identify the correct keyword relevant to the search keyword. The recognition of the keyword is no longer enough but also needs to provide rich information that contextualizes the semantic search keyword. This research mainly concentrates on the semantic search engine model to perform exact and accurate identification in big data.

II. Literature Survey

In this section, various bigdata related semantic search optimization models have been discussed—semantic Web information tracking and content filtration are essential topics for future generations. The relationship between IT resources and users can be established by using artificial intelligence and machine learning models. Exploring web-based content is a critical and challenging task in the big data environment. The computing methodologies, artificial intelligence, and knowledge-based learning mechanisms can handle big data analytics [1]. The singular value decomposition is a concept to extract the scientific database depending on priority lexical matching. The latent semantic search is a critical study associated with massive records and significant data analytics documents. The automatic content retrieval mechanism is necessary for the users to extract the information from advanced browsers. The computational comprehensive-textual searching encoding model is presenting many answers to the user queries. The fold-in and fold-out mathematical computations are giving the solutions to significant data updating processes through SVD[2]. They are describing the hidden information collection, document classification, and summarization. The tree classification and estimation process belonging to machine learning technology is attaining a low probability of errors [3]. According to a survey of J. Bobadilla[4], recommended a parallel web searching model. This content-based filtering mechanism has been incorporated through the Internet of things; this method provides a better content classification for the development of future applications. In the past, this type of methodology has not been presented. This collaborative filtering and massive prediction can be performed through the social Internet of things mechanism. The knowledge-based recommendation system provides human expert suggestions and data science techniques to differentiate the automatic content verification in big data analytics. The content-based, collaborative, knowledge, and hypothesis recommendation model is very useful for semantic search in the big data platform. In [5], it has been mentioned that the degree of document indexing and record balancing is very complicated, and the N-semantic recommendation system is introduced.

S no	Author	Technique	Keypoint
1	J. S. Breese et al. [6]	Predictive Collaborative Algorithm for Content Analysis	In this work, an empirical study of predictive collaboration algorithm is proposed to differentiate the big data's content, but this is taking more delay to extract the information.
2	L. Breslau et al. [7]	Web Catching Distributive Algorithm	The big data-based web request and searching methodologies artifact is the information. In this Zipf, a distribution mechanism has been incorporated for semantic significant data observations. This model is facing short-term delays and complex operations.
3	R. Burke et al. [8]	User Modeling and Adaptive Interactions	In this work, an advanced user interaction model has been designed. It is the largest big data model to extract the verification from websites, but it is taking more response time.
4	M. Cha et al. [9]	User Search File Priority Model	In this work, YouTube-related semantic search work is analyzed through fundamental keyword concepts. It is a high latency time process to extract the information.
5	M. Cha et.al[10]	Large Scale Big Data Integration	A semantic search optimization model is implemented, but it has more complex operations. It is unable to extract high-end big data content and does not work for colossal record analysis.
6	C. L. Clarke et al. [11]	Information Filtering with Hypothesis Model	In this work, a useful high dense content model has been designed with a filtration approach. It is a significantly less accurate process to get bigdata information.
7	P. Cremonesi et.al [12]	Cross-Domain Recommender System	In data mining, colossal data is present. It cannot be easily recognized by the users. So in this research work, an adaptive SEO method is used to identify the records available in bigdata.
8	P. Cremonesi et.al [13]	Low Dimension based Random Projection Mechanism	In this work, an adaptive random projection-based content searching mechanism has been implemented for big data applications. Browsers cannot handle clustering data. It can be possible only with the semantic search optimization model.
9	S. Dasgupta et, al [14]	Random partition KNN technique	The randomizing KNN model is a high dimensional mechanism; Here, records' overlapping has been controlled through the probability query point technique. It is a very complex process to retrieve information from big data.

10	C. M. De Vries et al. [15]	The Parallel Streaming Mechanism for Web-Scale Applications	A web-scale application has been working on the content differentiation and recognition process. In this work, because of parallelism, high area information tracking is necessary, so much delay is happening.
11	S. Deerwester et al. [16]	LSA indexing model	The latent semantic search analysis is a very complex process to get the information from indeed documents. If indexing and Id are missing, then content segmentation and filtering are not possible.
12	W. B. Frakes et.al [17]	Information Data Structure Algorithm	The data-structured optimization models are very complex to implement in HTML language. The delay and complex operation can limit data structure techniques. It is a significant drawback of retrieval in information data structured modeling.
13	D. Glowacka et al. [18]	Open source Web Content Scientific Analysis	A scientific keyword manipulation and its digital information retrieval system is introduced for the semantic database model
14	D. Goldberg et al. [19]	Collaborative Wave Filtering Technique	In this work, an effective wave filtering model identifies a group database in big data analytics.
15	K. Hajebi et al. [20]	K Nearest Neighbor Algorithm	In an offline step, the algorithm produces a nearest neighbor graph and performs to start from a randomly sampled node of the map when asked for a central topic. It provided security with precision and high computational efficacy quantitatively.
16	N. Halkoet al. [21]	Matrix Decomposition	In this work, the low-rank factorization method identifies the semantic keywords from the bigdata framework.
17	M. Hall et al. [22]	Weka20114 database	The Weka is a bigdata platform to search content from the browser. In this model, the latent time is more compared to the usual searching.
18	J. He et al. [23]	Kernel Hashing Technique	A scalable optimized kernel hashing mechanism is a useful data technique to evaluate the information from servers. In this, due to low correlation, waiting time is increasing exponentially.
19	D. Huynh et al. [24]	Web Semantic Browser	Flexible information has been attained from a bigdata platform. In this, users are continually getting a response from the server related to queries.
20	V. Hyv¨onen et al. [25]	Semantic Web Browser	In this Semantic web browser, the keyword searching mechanism helps the record analysis to the user query.

The above literature survey concentrates on various significant data methods and their limitations. Many conventional, machine and deep learning models have been implemented, but that functionality is facing high latency.

III. Methodology

In this section, KVASIR, effective data web content is taken as the data set. A scalable, adaptive support vector machine (ASVM) method identifies the semantic keywords with an accurate search. The Internet consists of huge and excessive information; if the user wants any content from the above database, an effective filtering process is required; otherwise, high latency and mismatched content are faced. To improve the user experience, an effective content identification algorithm is necessary. A latent semantic search analysis has been proposed with the ASVM mechanism. A KVASIR semantic search system with ASVM provides an integrated system with proactive web service. Using Spark and

Hadoop, the practical applications are working effectively. An SVM is a Randomized machine support indexing system that analyzes millions of records. In this section, KVASIR architecture is designed with an ASVM as the core algorithm. Here, all semantic search solutions have been presented and challenged the earlier methods.

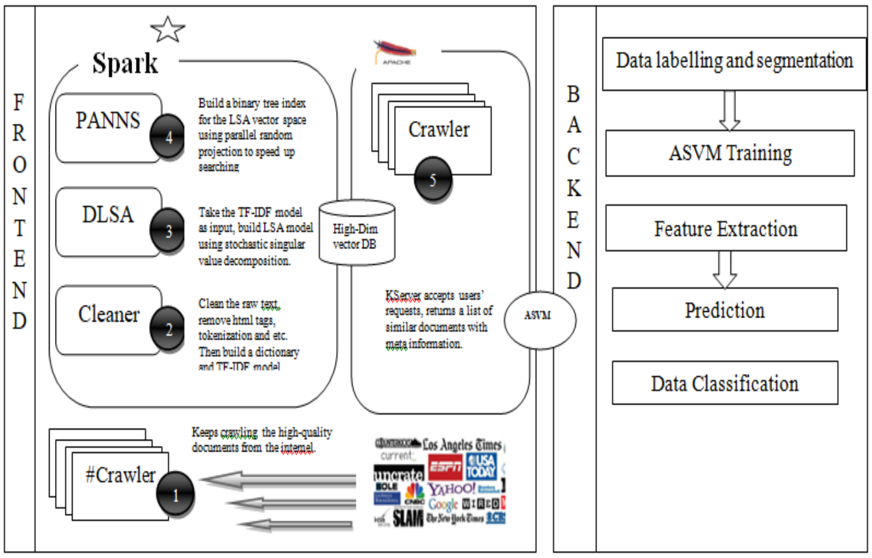

FRONTEND
Spark
PANNS
4
Build a binary tree index for the LSA vector space using parallel random projection to speed up searching
DLSA
3
Take the TF-IDF model as input, build LSA model using stochastic singular value decomposition.
Cleaner
2
Clean the raw text, remove html tags, tokenization and etc. Then build a dictionary and TF-IDF model
High-Dim vector DB
APACHE
Crawler
5
KServer accepts users' requests, returns a list of similar documents with meta information.
ASVM
#Crawler
1
Keeps crawling the high-quality documents from the internet.
BACKEND
Data labelling and segmentation
ASVM Training
Feature Extraction
Prediction
Data Classification

Fig 1. Proposed KVASIR with ASVM

Fig 1 clearly explains about proposed ASVM technique for future web browsers and big data analytics. The above architecture has been divided into two sections; in the first section, front-end tools are discussed, like PANNS, DLSA, CLEANER, CRAWLER, and K-Server. These five modules are operated in the front-end design. Coming to the second section, an extension browser is attached for searching content on the web page. The browser retrieves URLs to pages, the K-Server extension is running with the browser and simplifying the searching functionality.

Table 1. Database

database	#of entry	Raw_documents size	DocumentSize
Wiki Records	4.9×10^6	50.0 GB	AVG. 785 words
Flash News	5.6×10^5	1.8 GB	AVG. 650 words

Table 1 clearly explains the database of Wikipedia records and flash news from various international news channels. In this, 4.9 lakh entries and of size 50GBraw data has been collected for Wikipedia records. Moreover, 0.56 lakh entries with 1.8GB data and 650 average word length has been compiled for flash applications. Web servers and clouds consist of vast active and inactive information; therefore, content-based dynamic filtering is necessary for refining original data. In this work, a new semantic search mechanism has been implemented based on the adaptive SVM Machine Learning technique. Technologically advanced search methods like KVASIR for bigdata analytics give accurate results, but semantics search options, accuracy, recall, and F1-score parameters need improvement. This research work using ASVM-KVASIR architecture has been designed to improve the significant text content provision for big data sources (Internet). In ASVM, V-cross nonlinear randomization mechanism can differentiate the small margins and extensive margins of records with various scenarios such as date-year, area, and current application. This proposed architecture improves the V-cross classic randomized screen to maintain efficient searching and indexing for huge documents.

$$D = \left\{ (j, d_j) \middle| d_j = \frac{\Sigma_{\forall D_i} d_{i,j}}{\left(\Sigma_{\forall D_i \cdot} \left\| (j, D_i) \right)^3}, \forall (j, .) \in \cup_{\forall D_i} D_i \right\} \text{-----} (1)$$

Equation 1 explains about document indexing mechanism; here, D represents that indexing score j,d are indexing id and score, respectively. Using this expression, we are allocating the d score to KVASIR web browser content.

Table:2 Spark Database

No of CPU's	Time of cleaner	Time of DLSA	Time of PANNA's	Sum
5	1.62	21.26	14.21	37.09
10	0.34	6.79	2.91	10.04
15	0.21	4.52	1.23	5.96
20	0.23	3.23	0.97	4.43
25	0.42	2.42	0.83	3.67

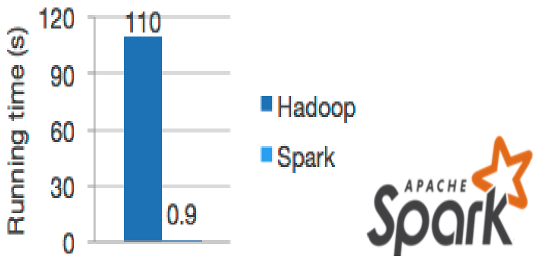


Fig 2 Sparkvs Hadoop

The Spark has used in KVASIR adaptive SVM system. It is a fast-unified analytic engine. The Spark system can be useful for large-scale data records processing in big data analytics. Hadoop and Spark are two running big data keyword searching builders. Compared to Hadoop, Spark systems are giving low response time, which is more useful in the big data platform. From the speed point of view, it is 100 times faster. Moreover, due to high-performance, batch and live streaming data have been scheduled with execution and query optimization. The spark system consists of three main blocks; PANNS, DLSA, CLEANER.

A. PANNS

This class builds up the index for the given data set. Two metrics are supported: Euclidean and Angular (cosine). The data set should be a matrix consisting of row vectors. For cosine, the data set has been assumed to be normalized where data has length 1. Load data set from an HDFS file. Proper care has to be considered for the performance of building up index as this may significantly degrade due to HDFS lookup overheads. Convert mtx and prj to map file to save memory space. It is very useful when dealing with a large dataset and parallel mode is activated. Skip if the data has already mapped

```
Z = spark.Panns.json("#logs.json#")
dZ.where("document > 2015")
select("name.first").disp()
```

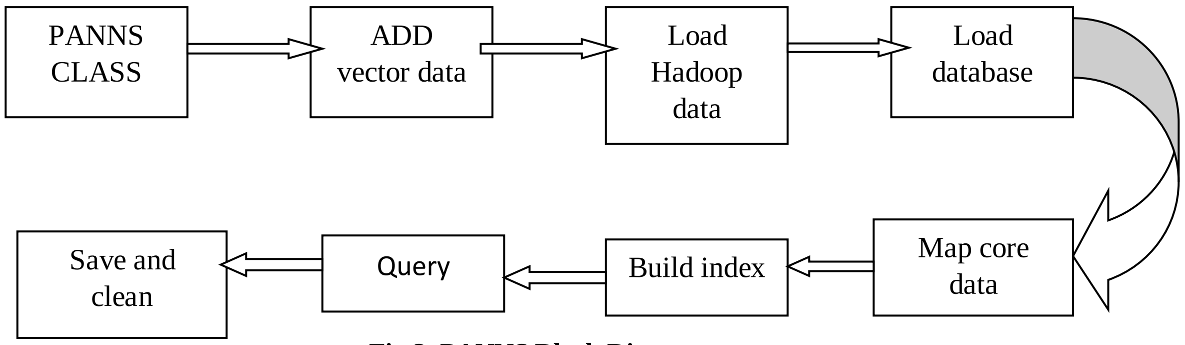


Fig 3. PANNS Block Diagram

Fig 3 clearly explains the PANNS operational process; according to the data set, it builds and provides the matrix normalization index. This PANNS offers dimensions for data according to Euclidean theory, and the data type is also assigned by using Euclidean angle theory. In the second step, vectors are listed out and load the query file to PANNS Hadoop block. The complete database is now loaded and transfer to the map-core block. This block saves the memory and deals with the parallel mode on available records using skip and unskip modeling. After this, the complete model has to be built using the random index multi-process mechanism. The PANNS model accepts the last query with an approximate vector score identifying the Browser's relevant data. It is saved in the tensor window until the calling function is inactive;finally the clean and save block performs the storage functionality with the f-frame self index mechanism.

B. DLSA

The distributive least-squares approximate (DLSA) is a local server for estimating the quadratic functions with global communication. It can easily approximate the 52GB data in 26ms using the conventional approach.

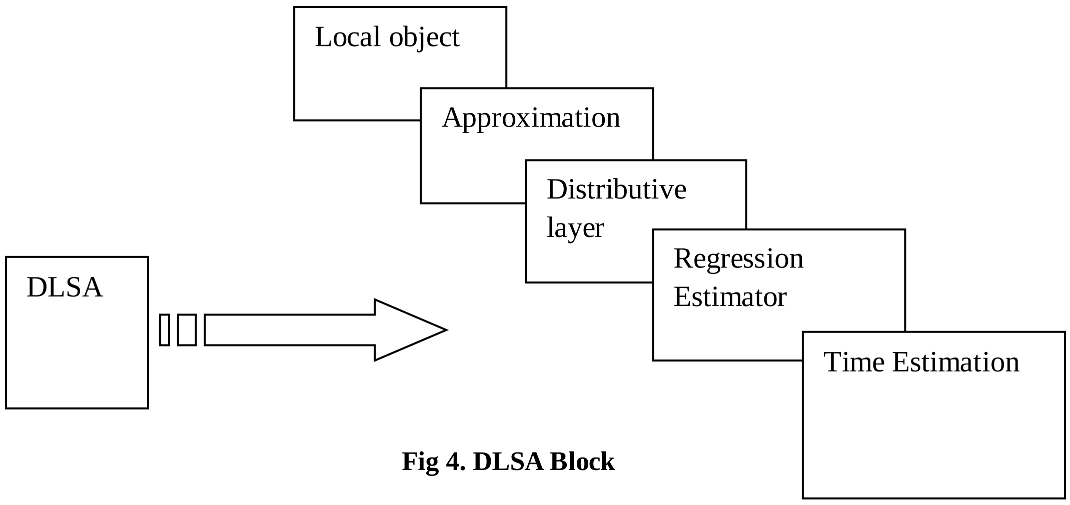


Fig 4. DLSA Block

```
setup(name='dlsa',

          use_scm_version=True,
          setup_requires=['setuptools_scm'],
          version='0.1.1',
          description='Distributed Least Squares
          Approximations',
          keywords='spark, spark-ml, pyspark, mapreduce',

          long_description=read('README.md'),
          long_description_content_type='text/markdown',

          url='https://wiki.com/feng-li/dlsa',
                              Data example author='Feng Li',
```

Fig 4 clearly explains VLSI functionality. In this, local objects and approximations collect the requirement from the map-reduce block. According to keyword search, distributive layers collect the information and send the response to the estimator block; in the final step, the time estimation function calculates the record uploading date.

C. Cleaner

The cleaner is an effective interaction block that allows directory directions according to the file location. Here file reading, file destination directories are cleaned by cleaner package and reducing the destination space.

The cleaner is a small entity function that can point out the irrelevant columns and convert the unstructured data into structured data. The structured data has been transformed into a TF-IDF vector format; it is a

Speed operated inbuilt logging operation in the spark block. The main functions of the cleaner are illustrating below.

- Alphanumeric and space score analysis with the linear function
- Read the multiple spaces and underscore characters
- Converting the whole paragraph into lower cases
- Removing the white spaces and leading to bugs

The above all functionalities have been maintained by a cleaner module in python software, and this facility does not identify in the Hadoop system.

D. Crawler

The crawler is dealing with a web package in which complete packages have been maintained by HTML and XML documents. The extracted data is easily analyzed through crawler block and providing the web decision. The Spark architecture crawler supports essential functions like removing the cheat records and restricting open-source files, and these can be extracted through web decision-making function.

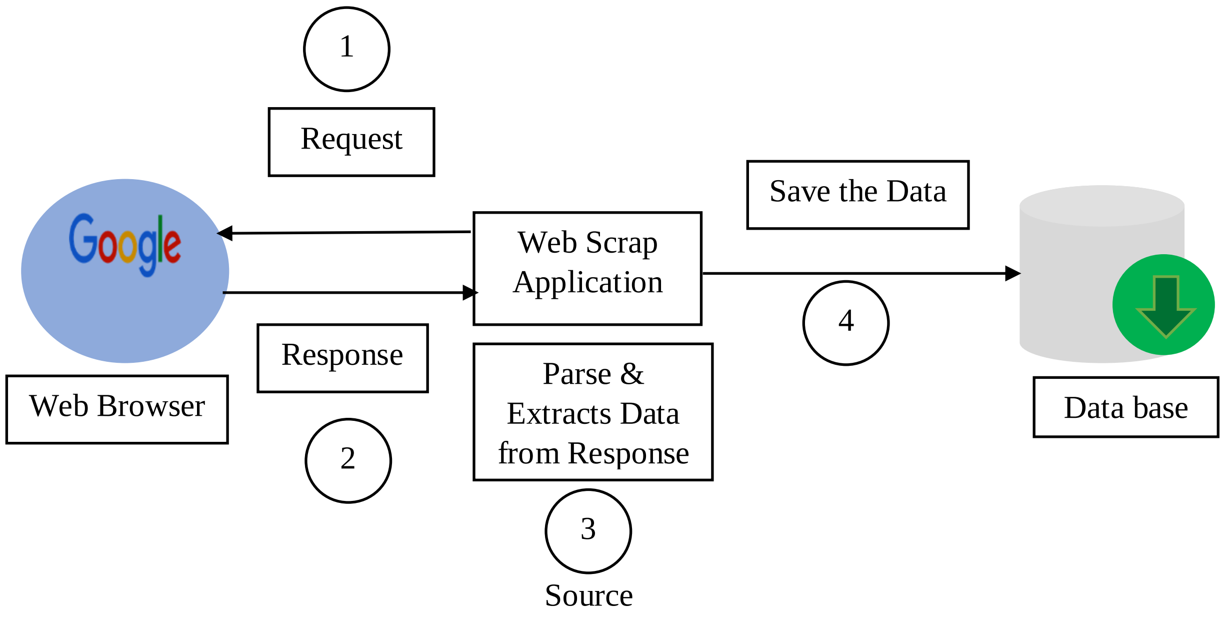


Fig. 5 Crawler Functional Block

The web Crawler is known as a spider bot, and it offers Internet service to browsers effectively. The browsers are utilized for crawling software to update the database continuously. Search engines, via crawler, copies the content and provides the indexes to selected records. The crawler's main blocks are scheduler, queue, multi-thread downloader, web page, text, and URL's. The following blocks help the content classification for big data analytics. It includes the scheduler block, multi-thread block, queue, and storage unit.

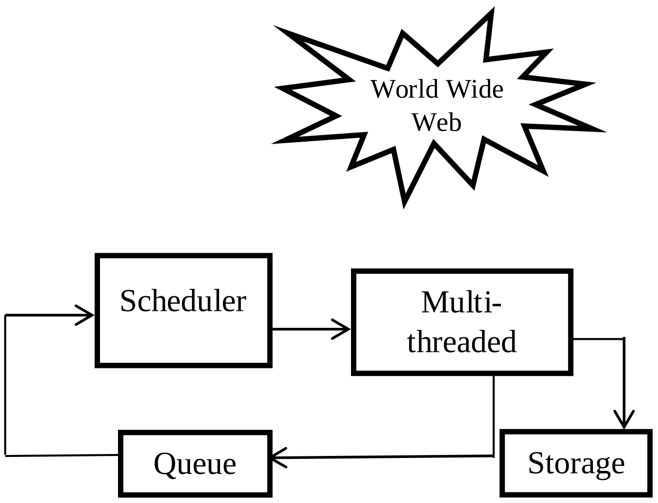


Fig 6. Crawler Flow Diagram

To stop crawling a specific resource more than once, crawlers typically perform some URL normalization.

URL normalization, also called canonicalization of URLs, refers to consistent updation and optimization of URL. Many forms of normalization can be done, including transformation to lowercase URLs, elimination of segments "." and ".." and adding trailing slashes to the non-empty path portion Fig6.

Above all, Spark and Hadoop blocks simplified the document differentiation and provided the indexing. This process has been performed in front-end design. In the next stage, the adaptive SVM model predicts and classifies the semantic data quickly.

E. Adaptive SVM Based Semantic Content Classification

In the back end-stage, an adaptive support vector machine has been applied for semantic content extraction. The earlier models, like PSO, GA, and KNN models, cannot classify the latest content through the selected browser. Therefore, to cross over the above limitations, a multi-class adaptive SVM model is proposed.

Algorithm: ASVM for content classification

Step 1: Adaptive multi-class SVM model is more useful in content classification with a decision function

$class\ of\ x \equiv argmax_{i=1,\dots,c}(w_1^T\varphi(x)+b_i)$ ------- (1)

The above equation 1is used to classify the data records and assign the class values according to regions' features. Here w is the weight of the form, b is the decision. If we apply the argument in the above functions, the class of semantic search can be obtained.

Step 2:

This c(c-1)/2 is a binary SVM process, which can differentiate two classes using score and accuracy. For the evaluation process, c-1 times the SVM classifier has been applied to extract the record score. Here k & j are classes using for training purposes, and (x,t) is known as a constraint for testing purposes.

$(w_{kj}^T\varphi(x_i)+b_{kj}) \geq 1-\xi_{kj}^t, for y_t = k,$ ------- (2)

$(w_{kj}^T\varphi(x_i)+b_{kj}) \leq -1+\xi_{kj}^t, for y_t = j,$ -------- (3)

$\xi_{kj}^t \geq 0.$---------- (4)

$w_{y_i}^T\varphi(x_i)+b_{y_i} \geq w_m^T\varphi(x_i)+b_m+2-\xi_i^m, \xi_j^m \geq 0.$

Here k, j classes are assigned for training purposes, and (x,y) are used for testing purposes. The multi-class SVM identifies the objective function and optimizes the decision by using equations 2,3&4.

$4(c-1)N^3/c^2$------------- (5)

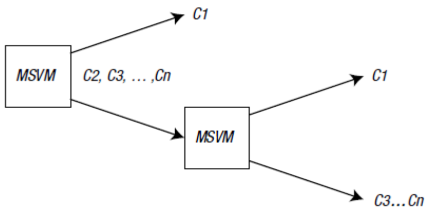


Fig 7: Confusion matrix

Equation 5 demonstrates the normalization of class according to adaptive SVM extraction. cN^3 represents the learning data that has been acquired from labeling and segmentation. Here various types are assigned for input to adaptive multi-class SVM technique. According to the weight balancing process and priority index, ID-based classes are differentiated. The following method can help the semantic search contents from a big data platform.

Step 3:

$x, \sum_{i=1}^{M} P(c_i/x) = 1,$ -------------------- (6)

$P_c = \sum_{i=1}^{M} P(x \in R_i, c_i) = \sum_{i=1}^{M} P(c_i)\int_{R_i} p(x/c_i)dx,$ -- (7)

Equation 6 & 7 clearly explains about multi-class SVM vector analysis using statistical information. If added, all classes are available in the data set; it can give the one as output. Here P(Ci/X)>m, then only it is considered as a perfect class. If this value is less than one, the probability of classification gets false errors.

$P_c = \sum_{i=1}^{M}\int_{R_i} p(x/c_i)p(x)dx \geq \frac{1}{M}\sum_{i=1}^{M}\int_{R_i} p(x)dx,$ ---- (8)

$\Rightarrow P_c \geq \frac{1}{M};$ ------(9)

In equation 7, P$_c$represents the probability of correct classification; in this, R$_i$ represents the region with feature decision in favor of C$_i$. Eq 8&9 is used to classify the records with statistical values. If the condition of equations is not satisfied, then the multi-class error is verified. Eq 10 briefly explains multi-class probability error with fault matching records.

$P_e = 1-P_c \leq 1-\frac{1}{M} = \frac{M-1}{M}$ ------- (10)

For a multiclassification flat formulation, P$_e$ increases with the number of categories M increases. The multiclassification task for a hierarchical model has simplified to discrete ones at each point, with P$_e$ $=\frac{1}{2}$ Therefore, the total error has been predicted to converge asymptotically to a lesser value for the clustered task than for a flat multiclassification method. In this step, any classification error is more than 0.5, then automatically, step 1 & step 2 have been performed for better semantic content classification

Step 4:

In this step, the performance measures are calculated using the confusion matrix. It is a probabilistic methodology used to identify unsupervised learning content. The adaptive SVM mainly concentrates on imbalanced data, imbalanced class, and non-synthetic data. The following data is known as unstructured data; these cannot be easily identified by previous semantic algorithms like KNN, PSO, and GA classification techniques. This is a significant challenge, and the risk of semantic content is very high. The unstructured data with earlier models are providing a higher lossy class index. So the classification and index allocation are not possible to generate the TF-IDF operation.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad \text{-------} \ (11)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad \text{-------} \ (12)$$

$$Specificity = \frac{TN}{TN+FP} \quad \text{-------} \ (13)$$

S.No	Query	True Positive	True Negative	False Positive	False Negative
1	Ireland Votes To Repeal Abortion Amendment dad shot	9.01	0.43	5.75	0.23
2	Singapore country business information	2.90	0.22	7.88	0.15
3	gold rates are varying continuously, reasons	3.51	0.48	3.16	0.14
4	world famous cricket batsman and his score	9.18	0.41	0.27	0.20
5	World wise best marketing country in consumer goods	3.31	0.28	5.73	0.14

When dealing with unstructured data, earlier semantic algorithms have been generating less accuracy. So the confusion matrix is generating a low-grade real positive rate, false-positive rate results. In this step, an adaptive feature extraction and classification SVM has been applied to improve the classification performance. The main objective of this work is illustrated using the following parameters:

1. Searcher intent from bigdata
2. Query context.
3. Keyword relationship.

The above three goals are designed by using python, HTML, and JAVA platforms.

Table 3 demonstrates that spark functionality, in this cleaner, DLSA, and PANNS tools are discussed briefly. This extension is verified on various CPUs starting from 5 to 40.

(c,t)		(20,16)	(20, 15)	(20,68)	(20,125)	(20,259)	(80,16)	(80,38)	(80,72)	(80,132)	(80,262)
Index_(MB)		371	735	1451	28955	5791	262	521	1035	2072	4152
exactness(%)		88.5	95.2	94.7	99.4	94.9	89.3	93.6	97.2	98.6	99.8
α_1=1.0	ms	2.4	3.5	4.2	5.7	6.5	4.2	7.6	11.0	13.3	15.8
α_2=0.1	ms	3.3	4.2	5.8	6.5	7.4	6.7	9.2	14.6	15.0	16.7
α_3=0.9	ms	4.2	4.5	6.5	7.6	8.1	8.8	11.5	14.9	17.1	17.5
α_3=0.7	ms	5.4	6.1	7.1	8.2	9.0	11.4	13.1	15.8	16.8	18.3
α_4=0.6	ms	6.0	6.3	7.5	8.6	9.4	13.5	15.7	18.2	19.5	20.8
α_5=0.3	ms	6.5	7.1	8.0	8.7	10.0	16.2	17.6	19.7	20.2	22.8

Above table 3 clearly explains document indexing and their scores using front-end and back-end modules. Now using the various CPUs with the ASVM RP-tree technique, we can generate the TF-IDF frames. After an extensive and in-depth classification process, we get the index ID and score. According to this, we classify the semantic content from the big data efficiently. In some instances, like as 20*64=80*16=1280. In this case, (20,64) accuracy is higher than (80, 16), i.e., 94.8 vs. 75.6. The following process is performed continuously until the proper classification cluster has been attained with more score and accuracy.

Table 4: Database for the experiment.

Rank	Page name	Source
#01	Cricket dataset	Wikipedia
#02	Corona update	Wikipedia
#03	Gold rate websites	CNN
#4	Marketing techniques	Wikipedia
#5	IT services	Wikipedia
#6	Online products	Thomson Reuters
#7	Fashion	CNN
#8	Country developments	BBC
#9	New technology	Markets Media
#10	University	Markets Media

Table 4 clearly explains various databases collected from popular websites. Using these records, we can handle the semantic data using the KVASIR ASVM methodology.

IV. Experimental Setup

In this research work, a semantic web content browser extension is designed for big data analysis for this experiment; drive HQ cloud is selected for CPU assignment. The dataset consists of 5 lakh clusters; each cluster size is 1kb. The drive HQ cloud economy is moderate for 100GB. In this cloud, we are dividing the clusters according to CPU assignment. For example, for 1CPU, 1 lakh clusters are assigned, and the process is going on.

V. Experimental Results

Kvasir: Scalable Provision of Semantically Relevant Web Content on Big Data Framework

On the internet, lots of information data gather and get relevant data for the recommendation. The author describes the semantic recommendation system based on latent semantic analysis (LSA means data will be retrieved based on semantic meaning). To search huge data author is using the below components to implement the KVASIR technique.

1) Crawler: we can crawl data from the internet or can upload a dataset to the application, and in this application, we are using NEWS dataset
2) Cleaning: This module accepts unstructured text data as input and then cleans it, and then converts input data into TF_IDF vector.
3) DLSA: This module converts the TF-IDF vector into the latent semantic analysis, and the author is using a stochastic SVD SPARK based algorithm to convert TF-IDF into an LSA vector.
4) PANNS: This module builds the search index from the DLSA vector, and to minimize memory usage, it will convert the search index into RP-tree, and this search index can be efficiently scanned using the KNN algorithm. This RP-tree will be created using SPARK parallel processing.
5) Query Search: This module accepts a query from the user and then performs a search operation using PANNS and KNN algorithm to obtained relevant data from RP-tree.
6) Adaptive SVM: This algorithm combines PANNS and SVM that accept input query and then find only top related relevant documents to query. Thus, the superior maximum appropriate document accuracy of this algorithm will be higher than PANNS.

Table 5: Comparison of results

	KvasirKNN[31]			ENN [30]			Kernel Hashing			Proposed Kvasir ASVM		
	50GB	100GB	1TB	50GB	100GB	1TB	50GB	100GB	1TB	50GB	100GB	1TB
Accuracy	80.1	81.32	87.56	91.23	92.54	96.32	93.23	94.56	89.91	99.32	99.56	99.72
Precision	72.23	72.26	81.53	89.12	90.13	91.54	91.32	93.46	97.42	98.54	99.82	99.13
Recall	85.12	83.42	88.92	85.26	85.52	86.22	85.52	89.92	91.52	96.53	97.52	98.53
Throughput	85.52	87.12	89.13	77.54	72.56	89.13	91.23	92.54	97.52	99.53	99.72	99.67
Scalability	89.52	91.23	87.53	82.52	87.32	89.71	72.53	91.34	93.72	99.54	97.23	99.16

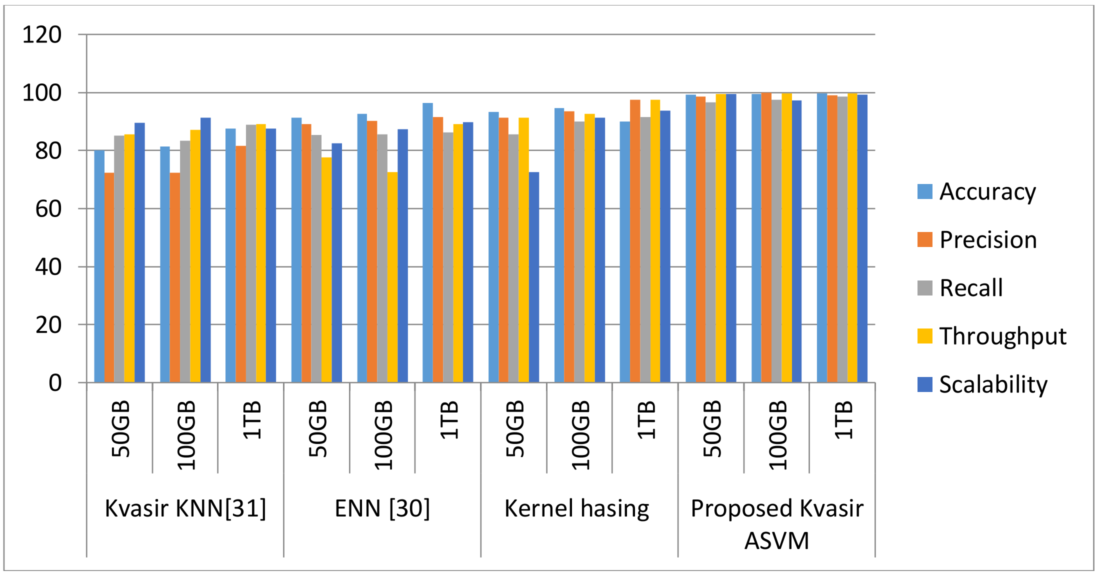


Fig 20. Comparison of Results

Table 5 and fig 20 clearly explain various models and their comparisons. In this, the proposed KVASIR adaptive SVM model attains more improvement compared to earlier methods.

VI. Conclusion

An advanced semantic search content classification application is designed for future big data analytics in this research work. The current technology is working on cloud and big data platforms, so users searching for these technologies' information is a complicated task. Due to robust data analysis, it requires an extension application for browsers. Therefore, an SVM machine with adaptive multi-level classification is provided as the solution for the above limitations. This work attains 0.97 average precision and 0.042 StdError. This means that the proposed methodology outperforms the experimental results and competes with the present models.

REFERENCES

[1] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. Scientific American, 284(5)(2001) 28–37.

[2] M. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. SIAM Review, 37(4)(1995) 573–595.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. J. Mach. Learn. Res., (2003) 3:993–1022.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez. Recommender systems survey. Knowledge-Based Systems, (2013).

[5] M. Brand. Fast low-rank modifications of the thin singular value decomposition. Linear Algebra and its Applications, (2006).

[6] J. S. Breese, D. Heckerman, and C. Kadie. The empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, (1998).

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In INFOCOM '99, IEEE, 1(1999) 126–134.

[8] R. Burke. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4)(2002) 331–370.

[9] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user-generated content video system. In ACM IMC'07, (2007).

[10] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn and S. Moon. Analyzing the video popularity characteristics of large-scale user-generated content systems. IEEE/ACM Trans. Netw., (2009).

[11] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. B¨ uttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In ACM SIGIR'08, (2008).

[12] P. Cremonesi, A. Tripodi, and R. Turrin. Cross-domain recommender systems. In Data Mining Workshops (ICDMW), IEEE, (2011).

[13] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In ACM Theory of Computing, (2008).

[14] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. CoRR, abs/1302.1948 (2013).

[15] C. M. De Vries, L. De Vine, S. Geva, and R. Nayak. Parallel streaming signature em-tree: A clustering algorithm for web-scale applications. In International Conference on World Wide Web, (2015).

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society for information science, 41(6) (1990) 391.

[17] W. B. Frakes and R. Baeza-Yates, editors. Information Retrieval: Data Structures and Algorithms. Prentice-Hall, Inc., USA, (1992).

[18] D. Glowacka, T. Ruotsalo, K. Konyushkova, K. Athukorala, S. Kaski, and G. Jacucci. Scent: A system for browsing scientific literature through keyword manipulation. In ACM International Conference on Intelligent User Interfaces Companion, (2013).

[19] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. Commun. ACM, 35(12)(1992) 61–70.

[20] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang. Fast approximate nearest-neighbor search with the k-nearest neighbor graph. In International Joint Conference on Artificial Intelligence, IJCAI'11 1312–1317. AAAI Press, (2011).

[21] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM, Rev., (2011).

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. ACM SIGKDD Explor. Newsl., 11(1)(2009) 10–18.

[23] J. He, W. Liu, and S.-F. Chang. Scalable similarity search with optimized kernel hashing. In ACM SIGKDD, (2010).

[24] D. Huynh, S. Mazzocchi, and D. Karger. Piggybank: Experience the semantic web inside your web browser. In Y. Gil, E. Motta, V. Benjamins, and M. Musen, editors, The Semantic Web, (2005).

[25] V. Hyv¨onen, T. Pitk¨anen, S. Tasoulis, L. Wang, T. Roos, and J. Corander. Technical report: Fast k-nn search. arXiv preprint arXiv:1509.06957, (2015).

[26] Joseph George, Dr. M.K Jeyakumar A Comparative Analysis of Data Integration and Business Intelligence Tools with an Emphasis on Healthcare Data International Journal of Engineering Trends and Technology 68.9(2020):5-9.

[27] Y. Jia, J. Wang, G. Zeng, H. Zha, and X.-S. Hua. Optimizing KD-trees for scalable visual descriptor indexing. In IEEE Computer Vision and Pattern Recognition (CVPR), (2010) 3392–3399.

[28] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, Recommender Systems Handbook, (2011) 145–186. Springer US,

[29] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In International Joint Conference on Artificial Intelligence, (2009).

[30] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In Advances in Neural Information Processing Systems, MIT Press, (2004).

[31] Wang, L., Tasoulis, S., Roos, T., &Kangasharju, J.., Kvasir: Scalable provision of semantically relevant web content on big data framework. IEEE Transactions on Big Data, 2(3)(2016) 219-233.

[32] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(1)(2011) 117–128.

Figures

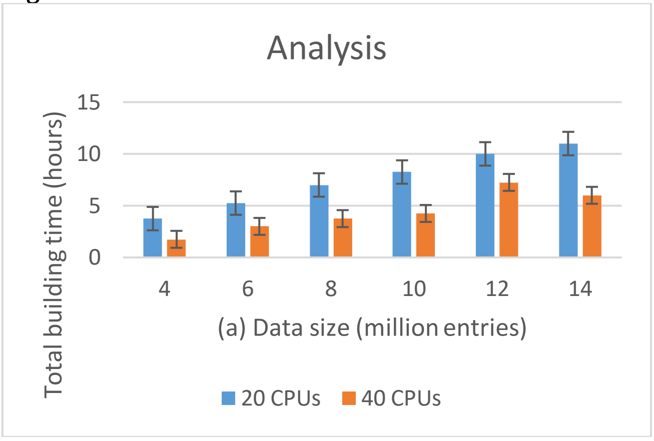
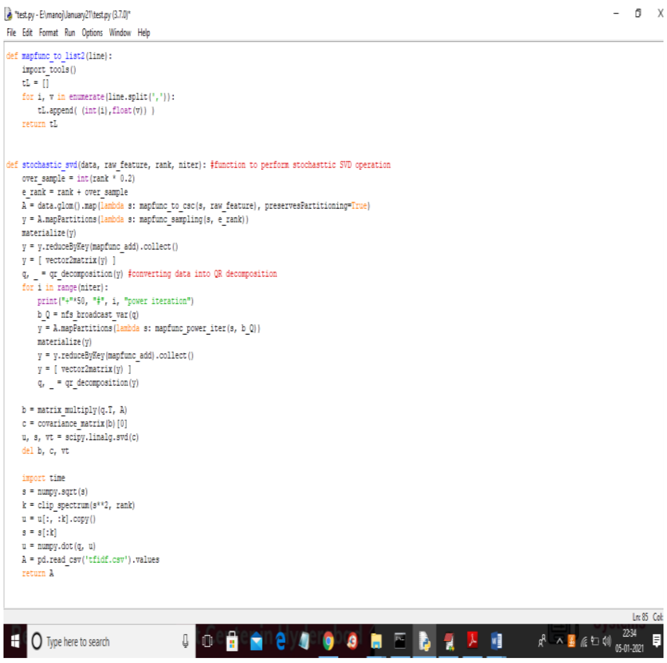


Figure 8: Data analysis using 20&40 CPU's.

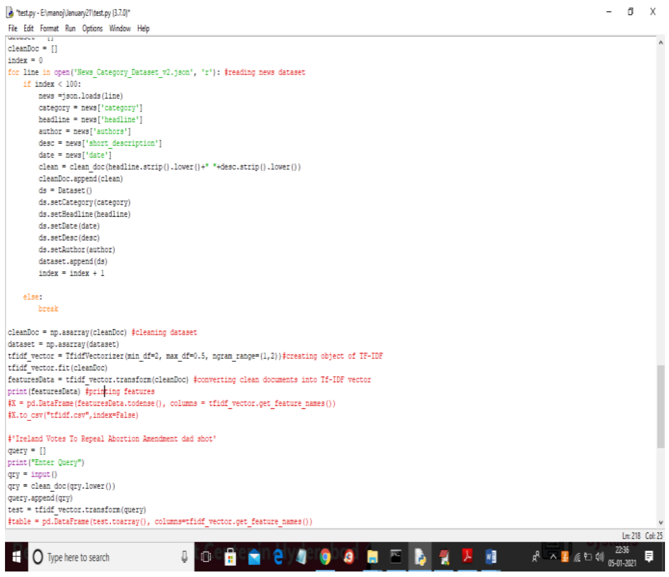
Fig 8 clearly explains about various CPU numbers taken for building the records with the browser. It has been observed that compared to 20CPU's, 40 CPUs easily search the data with less time, and this is a good achievement.


test.py - E:\manoj\January21\test.py (3.7.0)
File Edit Format Run Options Window Help
Type here to search

```
def mapfunc_to_list2(line):
    import_tools()
    tL = []
    for i, v in enumerate(line.split(',')):
        tL.append( (int(i),float(v)) )
    return tL


def stochastic_svd(data, raw_feature, rank, niter): #function to perform stochasttic SVD operation
    over_sample = int(rank * 0.2)
    e_rank = rank + over_sample
    A = data.glom().map(lambda s: mapfunc_to_csc(s, raw_feature), preservesPartitioning=True)
    y = A.mapPartitions(lambda s: mapfunc_sampling(s, e_rank))
    materialize(y)
    y = y.reduceByKey(mapfunc_add).collect()
    y = [ vector2matrix(y) ]
    q, _ = qr_decomposition(y) #converting data into QR decomposition
    for i in range(niter):
        print("+"*50, "#", i, "power iteration")
        b_Q = nfs_broadcast_var(q)
        y = A.mapPartitions(lambda s: mapfunc_power_iter(s, b_Q))
        materialize(y)
        y = y.reduceByKey(mapfunc_add).collect()
        y = [ vector2matrix(y) ]
        q, _ = qr_decomposition(y)

    b = matrix_multiply(q.T, A)
    c = covariance_matrix(b)[0]
    u, s, vt = scipy.linalg.svd(c)
    del b, c, vt

    import time
    s = numpy.sqrt(s)
    k = clip_spectrum(s**2, rank)
    u = u[:, :k].copy()
    s = s[:k]
    u = numpy.dot(q, u)
    A = pd.read_csv('tfidf.csv').values
    return A
```

Figure 9: DLSA operation.

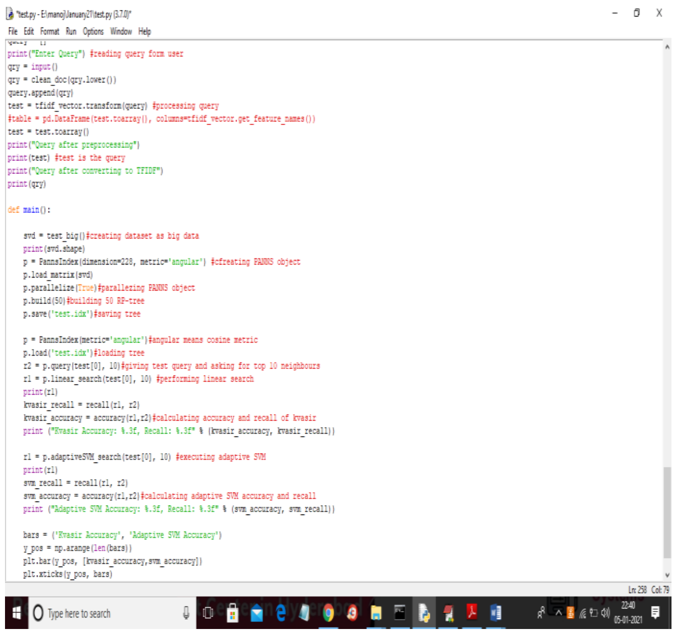
In below figure showing code to clean the dataset and to perform TF-IDF operation


test.py - E:\manoj\January21\test.py (3.7.0)
File Edit Format Run Options Window Help
Type here to search

```
cleanDoc = []
index = 0
for line in open('News_Category_Dataset_v2.json', 'r'): #reading news dataset
    if index < 100:
        news =json.loads(line)
        category = news['category']
        headline = news['headline']
        author = news['authors']
        desc = news['short_description']
        date = news['date']
        clean = clean_doc(headline.strip().lower()+" "+desc.strip().lower())
        cleanDoc.append(clean)
        ds = Dataset()
        ds.setCategory(category)
        ds.setHeadline(headline)
        ds.setDate(date)
        ds.setDesc(desc)
        ds.setAuthor(author)
        dataset.append(ds)
        index = index + 1

    else:
        break

cleanDoc = np.asarray(cleanDoc) #cleaning dataset
dataset = np.asarray(dataset)
tfidf_vector = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1,2))#creating object of TF-IDF
tfidf_vector.fit(cleanDoc)
featuresData = tfidf_vector.transform(cleanDoc) #converting clean documents into Tf-IDF vector
print(featuresData) #printing features
#X = pd.DataFrame(featuresData.todense(), columns = tfidf_vector.get_feature_names())
#X.to_csv("tfidf.csv",index=False)

#'Ireland Votes To Repeal Abortion Amendment dad shot'
query = []
print("Enter Query")
qry = input()
qry = clean_doc(qry.lower())
query.append(qry)
test = tfidf_vector.transform(query)
#table = pd.DataFrame(test.toarray(), columns=tfidf_vector.get_feature_names())
```

Figure 10: TF-IDF operation

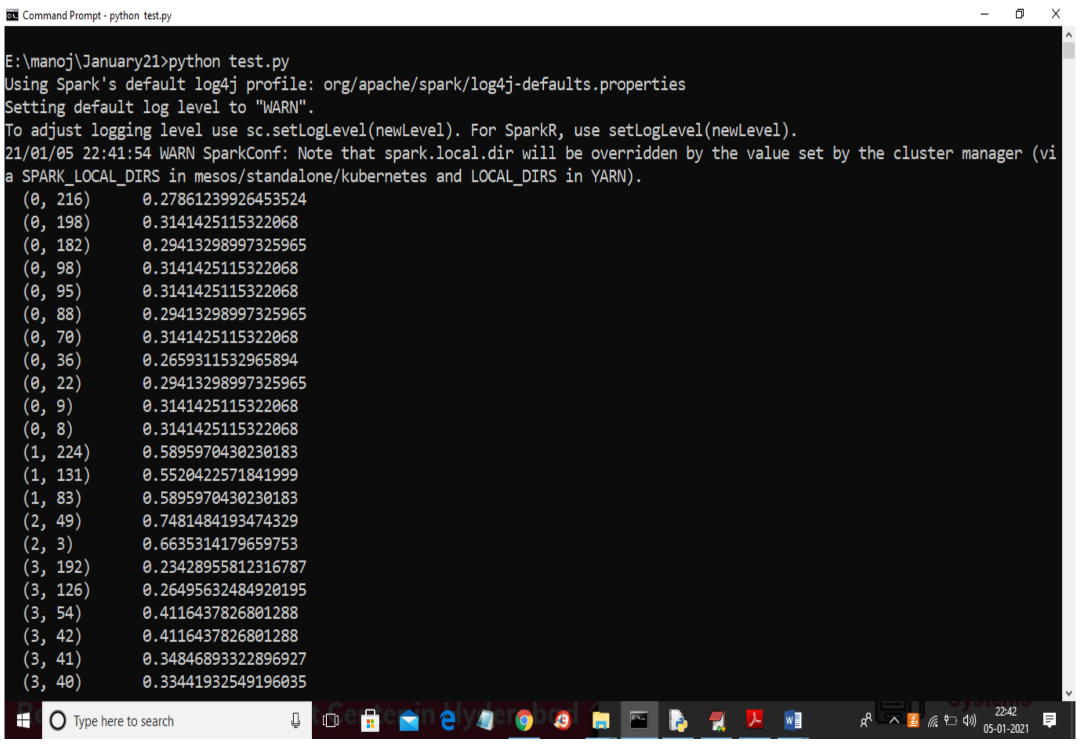
In below fig showing how to read the query from the user and then executing PANNS and Adaptive SVM algorithm to perform search operations and to get top 10 results with accuracy and recall


test.py - E:\manoj\January21\test.py (3.7.0)
File Edit Format Run Options Window Help
Type here to search

```
print("Enter Query") #reading query form user
qry = input()
qry = clean_doc(qry.lower())
query.append(qry)
test = tfidf_vector.transform(query) #processing query
#table = pd.DataFrame(test.toarray(), columns=tfidf_vector.get_feature_names())
test = test.toarray()
print("Query after preprocessing")
print(test) #test is the query
print("Query after converting to TFIDF")
print(qry)

def main():

    svd = test_big()#creating dataset as big data
    print(svd.shape)
    p = PannsIndex(dimension=228, metric='angular') #cfreating PANNS object
    p.load_matrix(svd)
    p.parallelize(True)#parallezing PANNS object
    p.build(50)#building 50 RP-tree
    p.save('test.idx')#saving tree

    p = PannsIndex(metric='angular')#angular means cosine metric
    p.load('test.idx')#loading tree
    r2 = p.query(test[0], 10)#giving test query and asking for top 10 neighbours
    r1 = p.linear_search(test[0], 10) #performing linear search
    print(r1)
    kvasir_recall = recall(r1, r2)
    kvasir_accuracy = accuracy(r1,r2)#calculating accuracy and recall of kvasir
    print ("Kvasir Accuracy: %.3f, Recall: %.3f" % (kvasir_accuracy, kvasir_recall))

    r1 = p.adaptiveSVM_search(test[0], 10) #executing adaptive SVM
    print(r1)
    svm_recall = recall(r1, r2)
    svm_accuracy = accuracy(r1,r2)#calculating adaptive SVM accuracy and recall
    print ("Adaptive SVM Accuracy: %.3f, Recall: %.3f" % (svm_accuracy, svm_recall))

    bars = ('Kvasir Accuracy', 'Adaptive SVM Accuracy')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, [kvasir_accuracy,svm_accuracy])
    plt.xticks(y_pos, bars)
```

Figure 11: PANNS adaptive SVM process

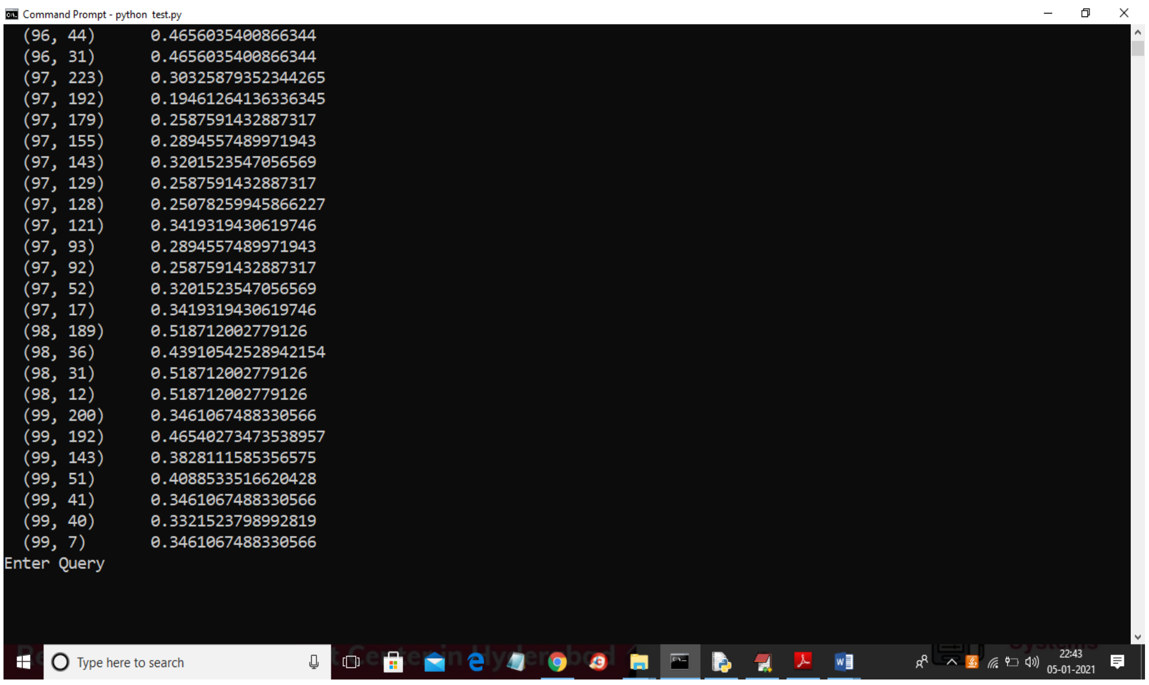
Now in the below figure, executing the above code as a console application. To run code, double click on the 'run.bat' file to get below output screen



```
E:\manoj\January21>python test.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/01/05 22:41:54 WARN SparkConf: Note that spark.local.dir will be overridden by the value set by the cluster manager (vi
a SPARK_LOCAL_DIRS in mesos/standalone/kubernetes and LOCAL_DIRS in YARN).
  (0, 216)      0.27861239926453524
  (0, 198)      0.3141425115322068
  (0, 182)      0.29413298997325965
  (0, 98)       0.3141425115322068
  (0, 95)       0.3141425115322068
  (0, 88)       0.29413298997325965
  (0, 70)       0.3141425115322068
  (0, 36)       0.2659311532965894
  (0, 22)       0.29413298997325965
  (0, 9)        0.3141425115322068
  (0, 8)        0.3141425115322068
  (1, 224)      0.5895970430230183
  (1, 131)      0.5520422571841999
  (1, 83)       0.5895970430230183
  (2, 49)       0.7481484193474329
  (2, 3)        0.6635314179659753
  (3, 192)      0.23428955812316787
  (3, 126)      0.26495632484920195
  (3, 54)       0.4116437826801288
  (3, 42)       0.4116437826801288
  (3, 41)       0.34846893322896927
  (3, 40)       0.33441932549196035
```

Figure 12: Index and score allocation

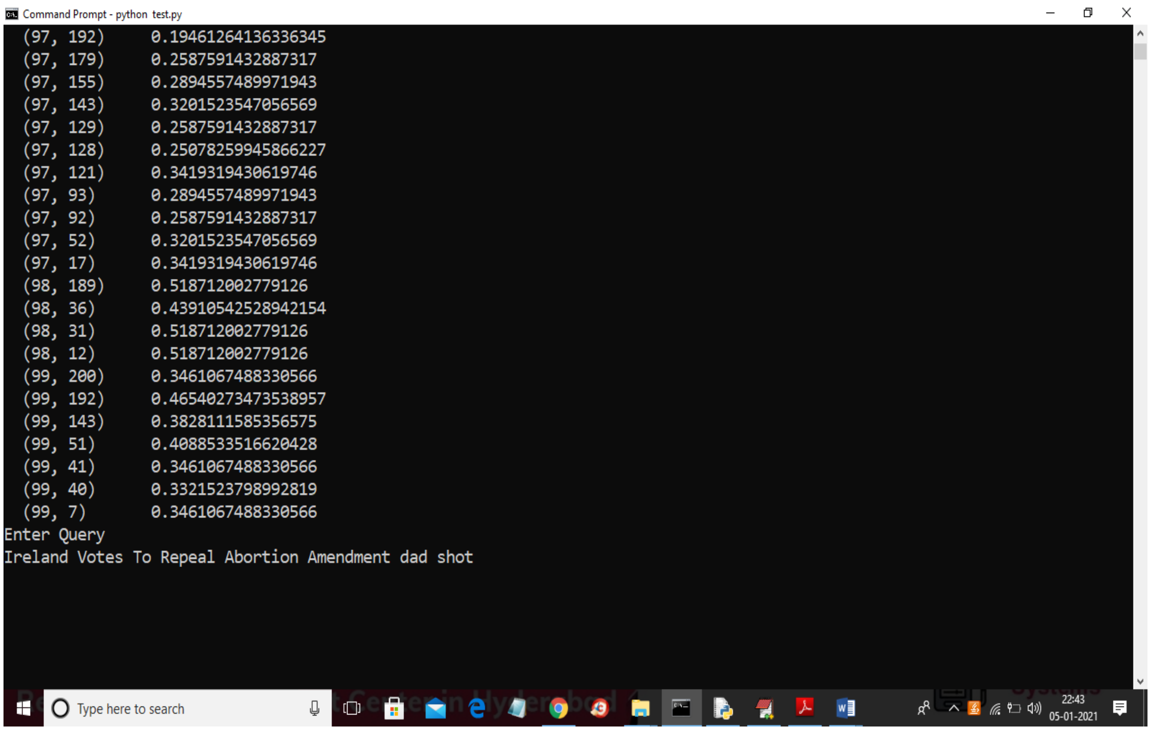
In the above figure executing test.py and then application read all dataset and then convert into TF-IDF vector, and then TF-IDF convert to DLSA array and then will get below the screen to enter the query



```
  (96, 44)      0.4656035400866344
  (96, 31)      0.4656035400866344
  (97, 223)     0.30325879352344265
  (97, 192)     0.19461264136336345
  (97, 179)     0.2587591432887317
  (97, 155)     0.2894557489971943
  (97, 143)     0.3201523547056569
  (97, 129)     0.2587591432887317
  (97, 128)     0.25078259945866227
  (97, 121)     0.3419319430619746
  (97, 93)      0.2894557489971943
  (97, 92)      0.2587591432887317
  (97, 52)      0.3201523547056569
  (97, 17)      0.3419319430619746
  (98, 189)     0.518712002779126
  (98, 36)      0.43910542528942154
  (98, 31)      0.518712002779126
  (98, 12)      0.518712002779126
  (99, 200)     0.3461067488330566
  (99, 192)     0.46540273473538957
  (99, 143)     0.3828111585356575
  (99, 51)      0.4088533516620428
  (99, 41)      0.3461067488330566
  (99, 40)      0.3321523798992819
  (99, 7)       0.3461067488330566
Enter Query
```

Figure 12: Query

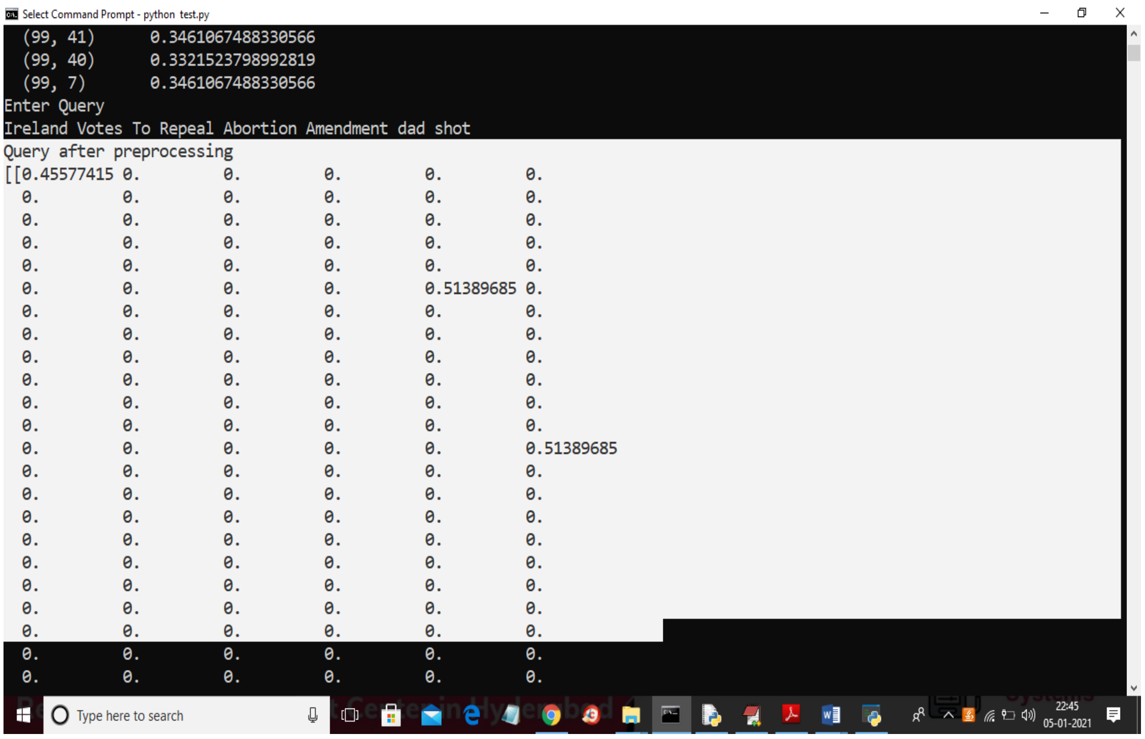
In the above fig enter query and then press enter key to get a result



```
  (97, 192)     0.19461264136336345
  (97, 179)     0.2587591432887317
  (97, 155)     0.2894557489971943
  (97, 143)     0.3201523547056569
  (97, 129)     0.2587591432887317
  (97, 128)     0.25078259945866227
  (97, 121)     0.3419319430619746
  (97, 93)      0.2894557489971943
  (97, 92)      0.2587591432887317
  (97, 52)      0.3201523547056569
  (97, 17)      0.3419319430619746
  (98, 189)     0.518712002779126
  (98, 36)      0.43910542528942154
  (98, 31)      0.518712002779126
  (98, 12)      0.518712002779126
  (99, 200)     0.3461067488330566
  (99, 192)     0.46540273473538957
  (99, 143)     0.3828111585356575
  (99, 51)      0.4088533516620428
  (99, 41)      0.3461067488330566
  (99, 40)      0.3321523798992819
  (99, 7)       0.3461067488330566
Enter Query
Ireland Votes To Repeal Abortion Amendment dad shot
```

Figure 13: Query and response

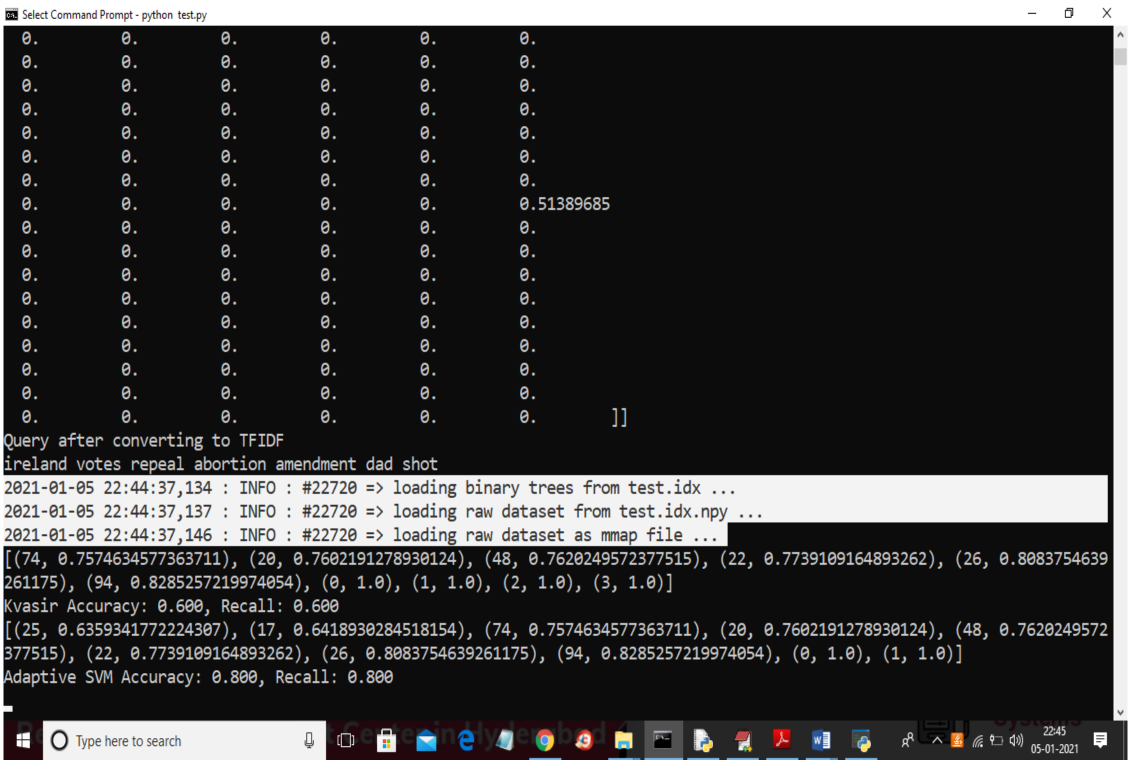
In the above fig, I entered query as 'Ireland Votes To Repeal Abortion Amendment dad shot' and then press the enter button to get the below result



```
  (99, 41)      0.3461067488330566
  (99, 40)      0.3321523798992819
  (99, 7)       0.3461067488330566
Enter Query
Ireland Votes To Repeal Abortion Amendment dad shot
Query after preprocessing
[[0.45577415 0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.51389685 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.51389685
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
```

Figure 14: Response

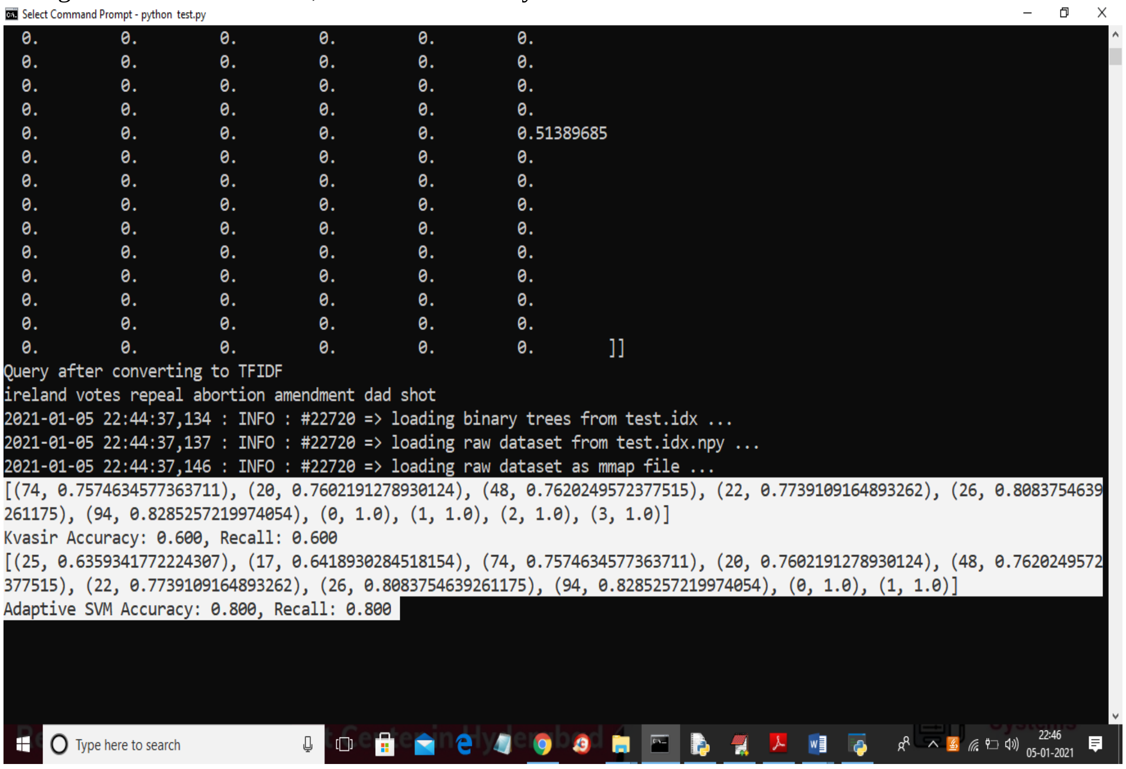
In the above fig query converted to DLSA TF-IDF vector and then will get below result



```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.51389685
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.        ]]
Query after converting to TFIDF
ireland votes repeal abortion amendment dad shot
2021-01-05 22:44:37,134 : INFO : #22720 => loading binary trees from test.idx ...
2021-01-05 22:44:37,137 : INFO : #22720 => loading raw dataset from test.idx.npy ...
2021-01-05 22:44:37,146 : INFO : #22720 => loading raw dataset as mmap file ...
[(74, 0.7574634577363711), (20, 0.7602191278930124), (48, 0.7620249572377515), (22, 0.7739109164893262), (26, 0.8083754639
261175), (94, 0.8285257219974054), (0, 1.0), (1, 1.0), (2, 1.0), (3, 1.0)]
Kvasir Accuracy: 0.600, Recall: 0.600
[(25, 0.6359341772224307), (17, 0.6418930284518154), (74, 0.7574634577363711), (20, 0.7602191278930124), (48, 0.7620249572
377515), (22, 0.7739109164893262), (26, 0.8083754639261175), (94, 0.8285257219974054), (0, 1.0), (1, 1.0)]
Adaptive SVM Accuracy: 0.800, Recall: 0.800
```

Figure 15: Semantic output

In the above figure in the selected text, we can see a binary RP tree loaded



```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.51389685
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.        ]]
Query after converting to TFIDF
ireland votes repeal abortion amendment dad shot
2021-01-05 22:44:37,134 : INFO : #22720 => loading binary trees from test.idx ...
2021-01-05 22:44:37,137 : INFO : #22720 => loading raw dataset from test.idx.npy ...
2021-01-05 22:44:37,146 : INFO : #22720 => loading raw dataset as mmap file ...
[(74, 0.7574634577363711), (20, 0.7602191278930124), (48, 0.7620249572377515), (22, 0.7739109164893262), (26, 0.8083754639
261175), (94, 0.8285257219974054), (0, 1.0), (1, 1.0), (2, 1.0), (3, 1.0)]
Kvasir Accuracy: 0.600, Recall: 0.600
[(25, 0.6359341772224307), (17, 0.6418930284518154), (74, 0.7574634577363711), (20, 0.7602191278930124), (48, 0.7620249572
377515), (22, 0.7739109164893262), (26, 0.8083754639261175), (94, 0.8285257219974054), (0, 1.0), (1, 1.0)]
Adaptive SVM Accuracy: 0.800, Recall: 0.800
```

Figure 16: Response with a semantic score

In the above figure, we can see the output of the top 10 search documents as a tuple; for example, in the above screen 74, the document id and 0.75 are the similarity value. In that array, we can see all 10 document search results. After that, we can see KVASIR accuracy as 0.60%. Then, we can see the search result of Adaptive SVM in the same tuple form; for example, 25 is the top document, and 0.63 is the cosine similarity searched by adaptive SVM. Then we can see SVM accuracy at 0.80%. Now below screen will show the accuracy of both algorithm in graph format

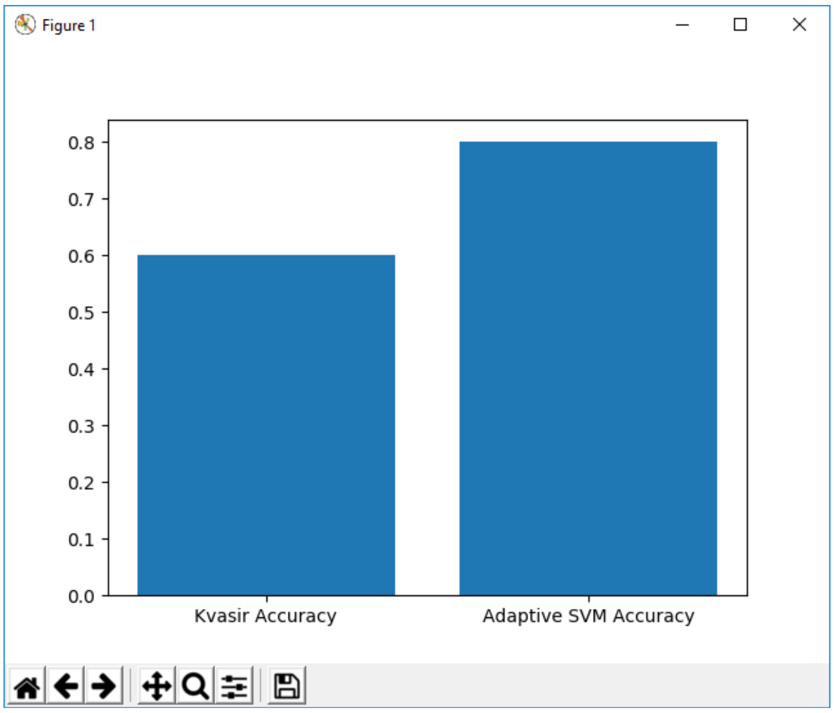
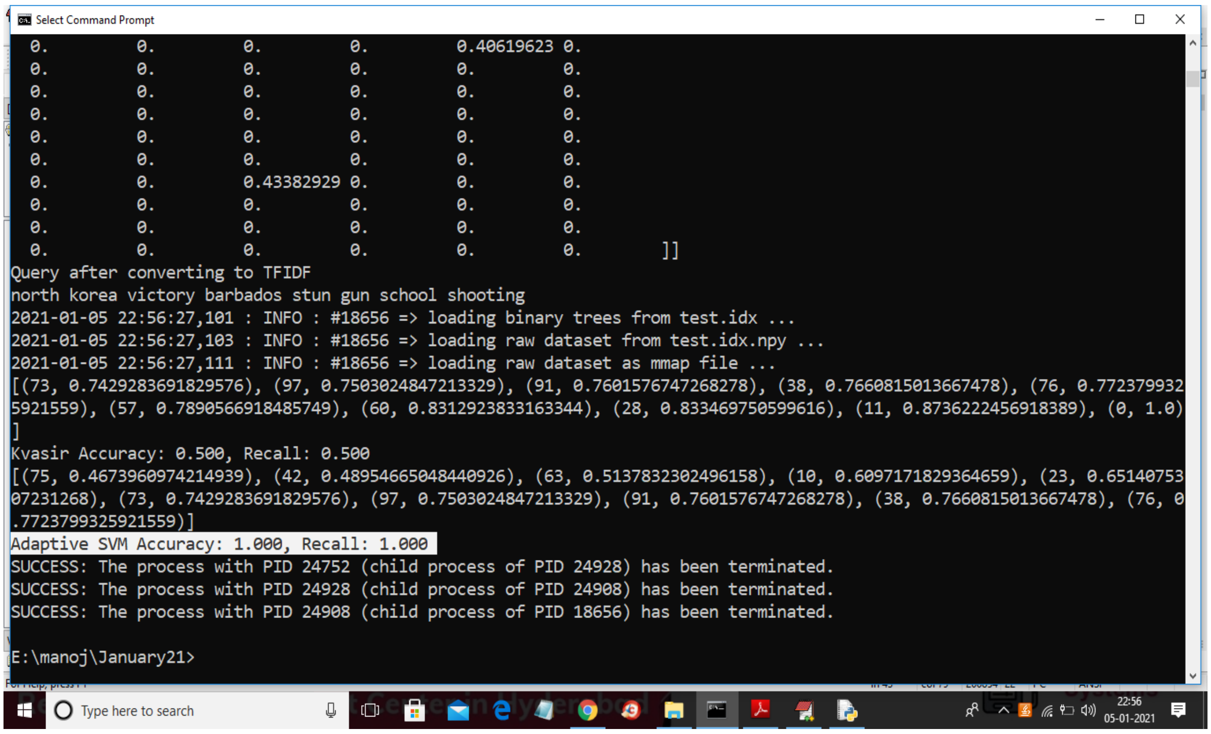


Figure 17: Accuracy

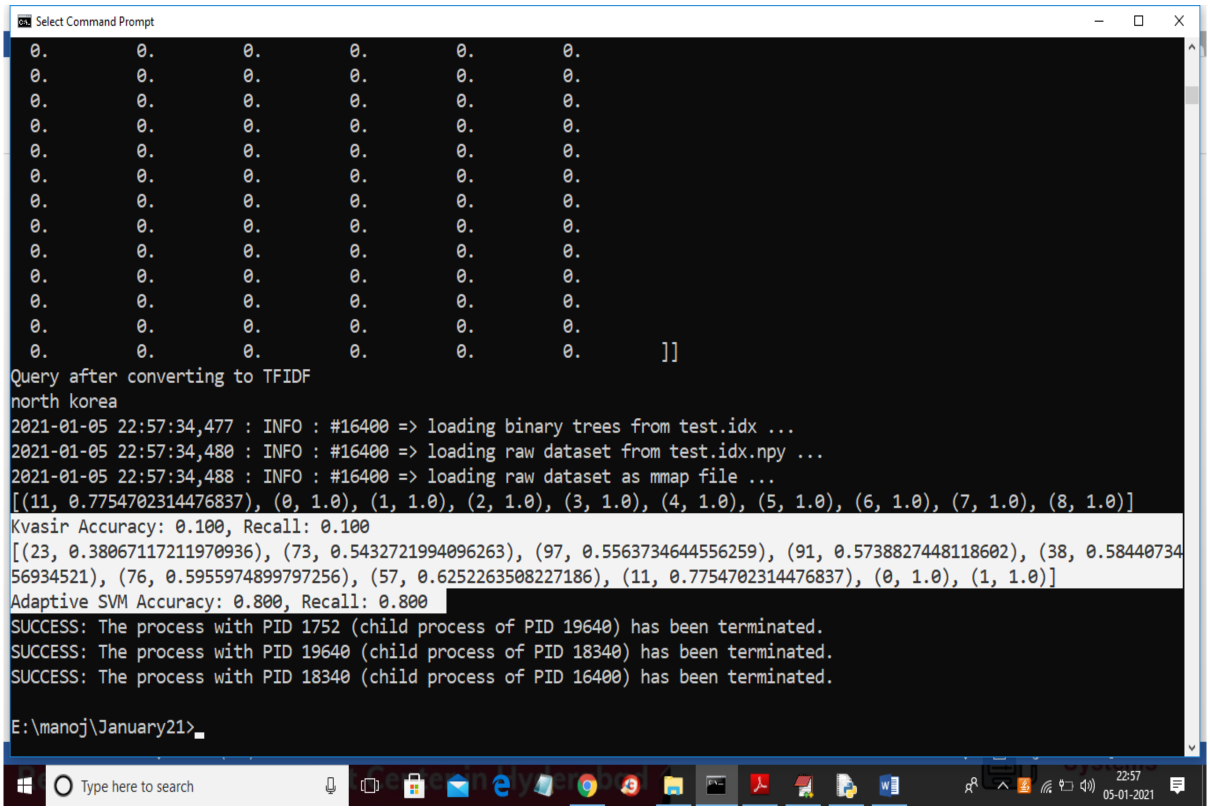
In the above figure, the x-axis represents the algorithm name, and the y-axis represents the accuracy of those algorithms. Sometimes for some queries, we may get accuracy as 100% also



```
 0.         0.         0.         0.         0.40619623 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.43382929 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.        ]]
Query after converting to TFIDF
north korea victory barbados stun gun school shooting
2021-01-05 22:56:27,101 : INFO : #18656 => loading binary trees from test.idx ...
2021-01-05 22:56:27,103 : INFO : #18656 => loading raw dataset from test.idx.npy ...
2021-01-05 22:56:27,111 : INFO : #18656 => loading raw dataset as mmap file ...
[(73, 0.7429283691829576), (97, 0.7503024847213329), (91, 0.7601576747268278), (38, 0.7660815013667478), (76, 0.772379932
5921559), (57, 0.7890566918485749), (60, 0.8312923833163344), (28, 0.833469750599616), (11, 0.8736222456918389), (0, 1.0)
]
Kvasir Accuracy: 0.500, Recall: 0.500
[(75, 0.4673960974214939), (42, 0.48954665048440926), (63, 0.5137832302496158), (10, 0.6097171829364659), (23, 0.65140753
07231268), (73, 0.7429283691829576), (97, 0.7503024847213329), (91, 0.7601576747268278), (38, 0.7660815013667478), (76, 0
.7723799325921559)]
Adaptive SVM Accuracy: 1.000, Recall: 1.000
SUCCESS: The process with PID 24752 (child process of PID 24928) has been terminated.
SUCCESS: The process with PID 24928 (child process of PID 24908) has been terminated.
SUCCESS: The process with PID 24908 (child process of PID 18656) has been terminated.

E:\manoj\January21>
```

22:56
05-01-2021

Figure 18: Accuracy and recall



```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         ]]
Query after converting to TFIDF
north korea
2021-01-05 22:57:34,477 : INFO : #16400 => loading binary trees from test.idx ...
2021-01-05 22:57:34,480 : INFO : #16400 => loading raw dataset from test.idx.npy ...
2021-01-05 22:57:34,488 : INFO : #16400 => loading raw dataset as mmap file ...
[(11, 0.7754702314476837), (0, 1.0), (1, 1.0), (2, 1.0), (3, 1.0), (4, 1.0), (5, 1.0), (6, 1.0), (7, 1.0), (8, 1.0)]
Kvasir Accuracy: 0.100, Recall: 0.100
[(23, 0.38067117211970936), (73, 0.5432721994096263), (97, 0.5563734644556259), (91, 0.5738827448118602), (38, 0.58440734
56934521), (76, 0.5955974899797256), (57, 0.6252263508227186), (11, 0.7754702314476837), (0, 1.0), (1, 1.0)]
Adaptive SVM Accuracy: 0.800, Recall: 0.800
SUCCESS: The process with PID 1752 (child process of PID 19640) has been terminated.
SUCCESS: The process with PID 19640 (child process of PID 18340) has been terminated.
SUCCESS: The process with PID 18340 (child process of PID 16400) has been terminated.

E:\manoj\January21>_
```

Figure 19: Performance measures

Similarly, you can enter any query and get the result of the query is found in the RP tree.

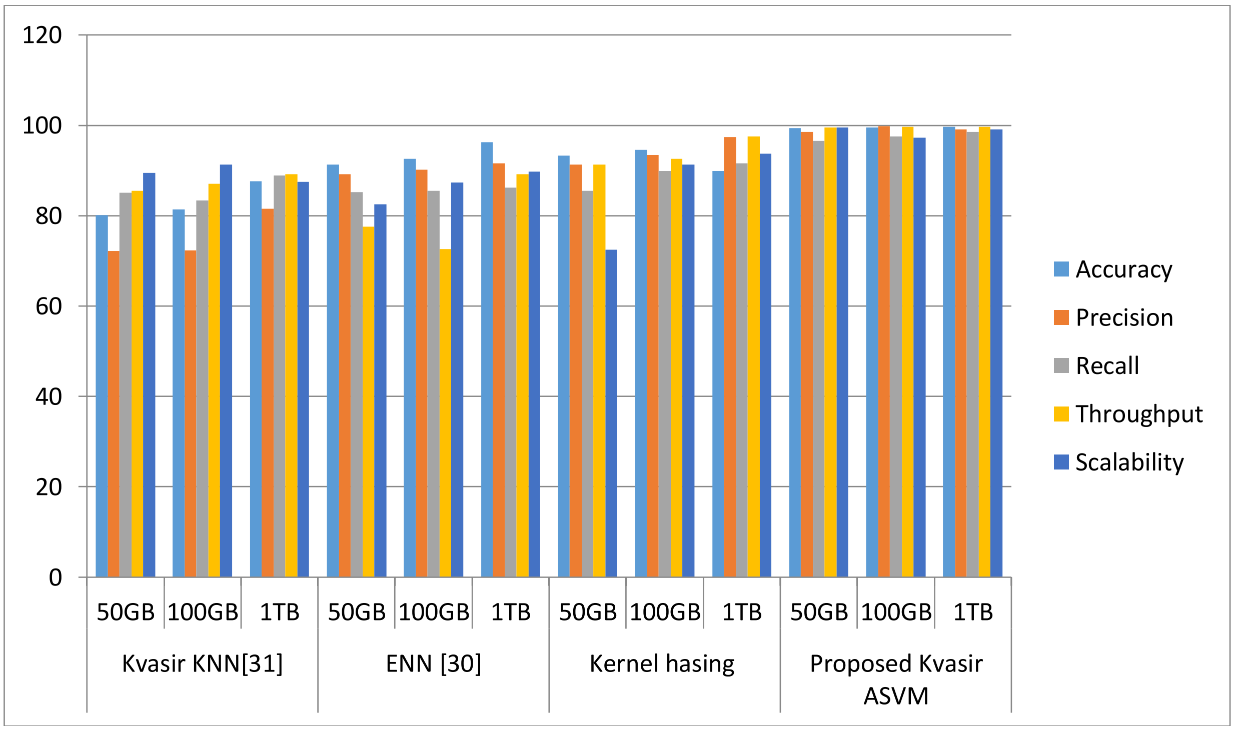


Figure 20: Comparison of results.

Table 6 and fig 20 clearly explain various models and their comparisons; in this, the proposed kvasir adaptive SVM models attain more improvement compared to earlier methods.