Biblio-Finder

Search Engine for Academic Conference Papers

Priyanka Motwani
M.Sc Computer Science
University of Windsor
Student Id: 105216601
Email: motwanip@uwindsor.ca

Vinay Kiran Manjunath
M.Sc Computer Science
University of Windsor
Student Id: 105158886
Email: manjunav@uwindsor.ca

***Abstract*—A search engine can be used for information extraction. We have proposed a system which retrieves the results having titles of academic conference papers and other useful information based on the keyword searched. In this project, we tried to implement few basic features that included Indexing, Searching with some additional features like Highlighting and SpellCheck. Moreover, features like PageRanking, Word2vec, DOc2vec, TF-IDF and Autosuggestion were also included as a part of this project. In addition to these features, classification algorithms like Naive Bayes, Logistic Regression and SVM along with clustering algorithms like K- Means have been implemented. Feature selection technique like Chi squared has been applied. The implementation has been done on five datasets that included conference papers from multiple resources. The mentioned features are implemented using Java Lucene library and other Python libraries. The proposed system is deployed on localhost using Apache Tomcat Server.**

***Index Terms*—Indexing, Searching, Spellcheck, Highlight, Lucene, Word2vec, PageRank, TF-IDF, Autocomplete, Autosuggest, Naive Bayes, Logistic Regression, K Means Clustering, Doc2vec, feature selection, SVM.**

I. Introduction

Information Retrieval Systems include digital documents with a framework, important text material and other media. A search engine is one of the approaches that suffice the implementation of IRS. Apache's Lucene library and its APIs have been used to build the features of this search engine. Other python libraries have also been used for the same. This search engine is built and enhanced with multiple features to search from a dataset having 3,079,007 research papers papers. Working of these features and algorithms have been analyzed in detail further. The implementation of all these features and algorithms have been applied to five different conference papers' dataset and evaluation has been done based on the results and performance metrics.

II. Search Engine Concepts

A. Search Engine

Search engine can be used for information extraction. It retrieves the data searched by the users and gives the results matched based on the keywords or terms entered by the user. Search engine indexing is the process of a search engine collecting, parses and stores data for use by the search engine. The specific index for the search engine is the location where all data obtained by the search engine are stored. It is the index of the search engine that provides the answers for search queries, and the links that are stored in the index of the search engine that appear on the results page. Search engine spiders, also called search engine crawlers, are how the search engine index gets its information, as well as keeping it up to date and free of spam.

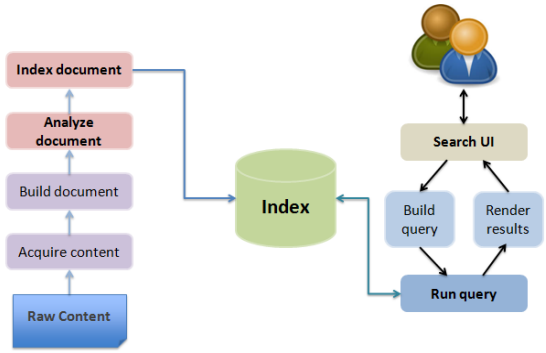

Fig 1. Basic Architecture of a Search Engine

B. Document

The word document refers to a single page or object in a search index for a website. For example, a document may be one single article on a publishing website.

C. Field

In the search environment, the term fields refers to the various components or elements in a search engine index that form the schema of a database. Different forms of documents cover various fields.

D. Keywords

Keywords are the words and phrases that Internet users type into a search engine's search box, like Google, to find the websites suit what they're searching for.

E. Query

A web search query is a query based on a particular search term that a user enters a web search engine in order to satisfy their needs for information. Web search queries are unique

in that they are mostly plain text or hypertext with optional search-directives.

F. Indexing

The way to eliminate scanning the texts sequentially for each query is by indexing the documents beforehand. Document indexing is one of the useful methods to assist with subsequent retrieval of records from repositories containing thousands of documents.

G. Searching

The indexed documents are searched in the repositories and the query returns the matched results. The assigned terms must reflect the content of the document to allow effective keyword searching.

H. Query Processing

A search engine's core functions may be defined as crawling, data mining, indexing, and query processing. Query processing is the mathematical process in which an individual's question is compared with the index and the results retrieved are presented accordingly.

III. Resources Used

A. Apache Lucene

Lucene is a powerful Java search library licensed under the liberal Apache Software that helps to add search feature to any application. Lucene provides a simple yet powerful core API that requires minimal understanding of full-text indexing and searching. Apache Lucene is an open source API that helps to search a large number of files.

B. Tomcat Server

Apache Tomcat is an application server built for executing Java servlets and rendering web pages using page coding on Java Server. Accessible either as a binary or as a source code version, Tomcat has been used to power a wide range of Internet applications and websites.

C. sklearn library

Scikit-learn (also known as sklearn) is a Python programming language free software machine learning library. This features various algorithms for classification, regression and clustering, including supporting vector machines, random forests, gradient boosting, k-means, and is designed to parallelize with the NumPy and SciPy numerical and scientific libraries.

D. Gensim

Gensim is an open-source library which uses advanced statistical machine learning for unsupervised topic modelling and natural language processing. Gensim is designed to manage large collections of text using data streaming and incremental online algorithms, which distinguishes it from most other machine learning software packages that only perform processing in memory.

E. NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

IV. Datasets Used

A. ICSE

The International Conference on Software Engineering comprises of all the papers that contain most recent innovation, trends, experiences and concerns in the field of software engineering.

B. VLDB

The International Conference on very Large Databases constitutes of research papers focusing on research in the field of database management

C. Sigmod

The International Conference on Management of Data contains research papers focusing on principles, techniques and applications of database management systems and data management technology.

D. Dataset Description

The distribution of the ICSE, Sigmod and VLDB dataset is in the following manner: docId, title, authors, year, conference, fulltext, citation, simdocs.

E. AMiner Network Citation Dataset

This Aminer Academic Social Networks intergrate publications from DBLP and citation links from ACM Digital Library, CiteSeer and other sources.

#* -- paperTitle	#@ -- Authors	#t -- Year	#c -- publication venue	#index 00---- index id of this paper	#% --- the id of references of this paper (there are	#! -- Abstract
#*Information geometry of U-Boost and Bregman divergence	#@Noboru Murata,Takashi Takenouchi,Takafumi Kanamori,Shinto Eguchi	#2004	#cNeural Computation	#index436405	#%94584 #%282290 #%605546 #%620759 #%564877 #%564235 #%594837 #%479177 #%586607	#!We aim at an extension of AdaBoost to U-Boost, in the paradigm to build a stronger classification machine from a set of weak learning machines. A geometric understanding of the Bregman divergence defined by a generic convex function U leads to the U-Boost method in the framework of information geometry extended to the space of the finite measures over a label set. We propose two versions of U-Boost learning algorithms by taking account of whether the domain is restricted to the space of probability functions. In the sequential step, we observe that the two adjacent and the initial classifiers are associated with a right triangle in the scale via the Bregman divergence, called the Pythagorean relation. This leads to a mild convergence property of the U-Boost algorithm as seen in the expectation-maximization algorithm. Statistical discussions for consistency and robustness elucidate the properties of the U-Boost methods based on a stochastic assumption for training data.

Figure . Aminer Dataset Structure

V. Added Features and their Implementation

A. Indexer and Searcher

Lucene's API is used for indexing and searching. Index Path is provided where the indexes are stored and Document Path is provided from where the documents are indexed. The following components are used:

a. IndexWriter: It is the central component of the indexing process. This class creates a new index or opens an existing one, and adds, removes, or updates documents in the index.

b. Standard Analyzer: It is the most generic analyzer as it can process and can tokenize all the lexical type. It also includes

features like stopwords removal and can differentiate different types of text and is a commonly used analyzer.
c. IndexSearcher: It is a gateway to index search. All searches come through an instance of IndexSearcher using any of the several methods of overloaded search.
d. QueryParser: Processes a human-entered (and readable) expression into a concrete Query object.

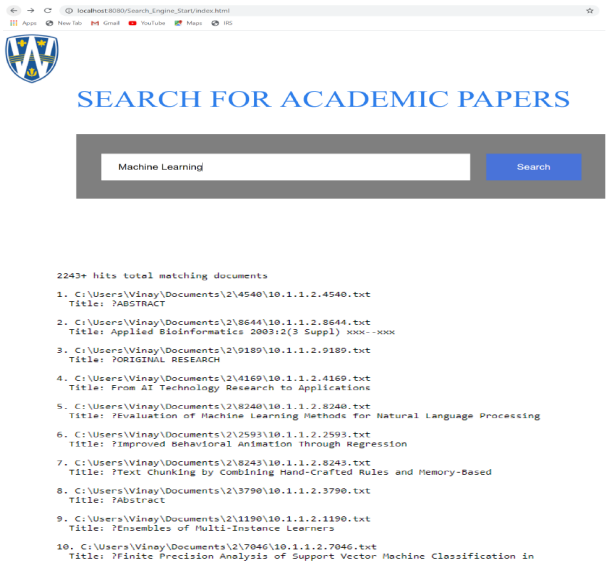

Figure 1. Results retrieved by the Search Engine

B. *Highlighting*

The highlighter module fragments and highlights the text based on the search query. Each hit includes some number of fragments of the matching document highlighting the terms of the query. The following components are used for highlighter:
a. QueryScorer: QueryScorer extracts matching spans for the query, then uses these spans to score each fragment. Fragments that didn't match the query, even if they contain a subset of the terms from the query, receive a score of 0.0.
b. Fragmenter: Fragmenter is a Java interface in the Highlighter package whose purpose is to split the original string into separate fragments for consideration.
c. Formatter: The Formatter Java interface takes each fragment of text as a String, as well as the terms to be highlighted, and renders the highlighting. SimpleHtmlFormatter is used.

C. *Spell Checking*

Spell Checking works by taking an existing field from the main index and creates a secondary index specifically designed to quickly look up suggestions for candidates.This index is constructed by creating n-grams of the words from the original field based on character.The word to be checked is properly analyzed at the time of query and then searched against this secondary index.The following components are used for Spellchecker:
KeyWordAnalyzer: Treats entire text as a single token.

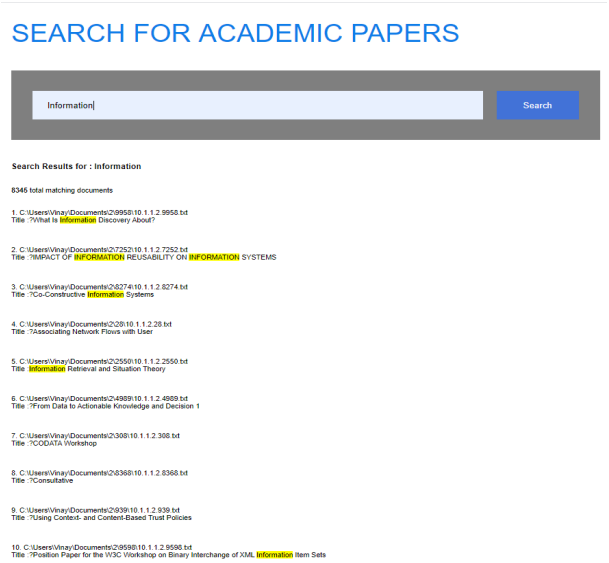

Figure 2. Results retrieved by the Search Engine using Highlighter

1) Ngram Distance: A letter ngram is all subsequences of adjacent letters in length, varying between a minimum and a maximum size. Using this method the ngrams are indexed into a separate spell checker database for all words in the dictionary. This is usually a quick procedure, so that the indexing phase of the application will restore the entire spell checker index whenever the main index is updated. It is used as a String Distance for SpellChecker.

Word	Lettuce	Letuce
3gram	let, ett, ttu, tuc, uce	let, etu, tuc, uce
4gram	lett, ettu, ttuc, tuce	letu, etuc, tuce

TABLE I. NGRAMS

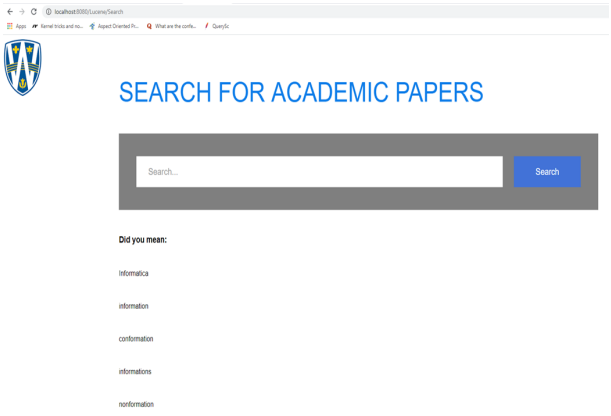

Figure . Putting all the points of into a matrix and plotting in 2D

D. *PageRank*

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites. It is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose

of ”measuring” its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The PageRank algorithm outputs a probability distribution used to represent the likelihood of a person randomly clicking on links will arrive at any particular page.

Implementation Steps:

- NetworkX Python package is used for calculating PageRank values and the algorithm is applied on AMiner Network Citation Dataset V1. Data is cleaned and page rank values are calculated.
- First, the distribution of the in-degree (number of citations) is shown below. It is almost linear on the log-log scale, which confirms the scale-free nature of the network.

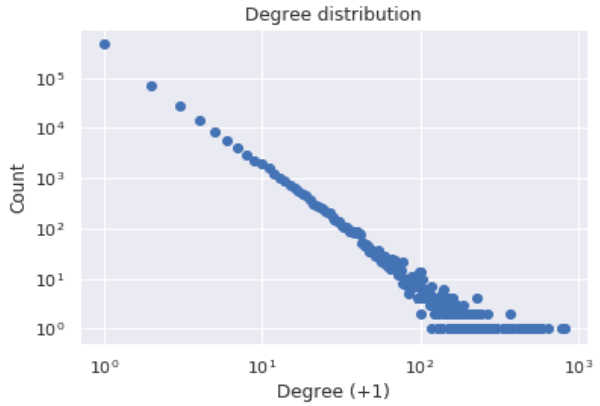

Figure . Putting all the points of into a matrix and plotting in 2D

- Next, the distribution of PageRank is shown below. Similarly, it is almost linear on the log-log scale.

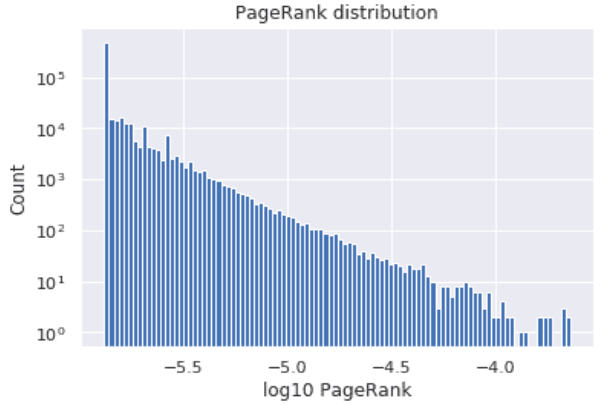

Figure . Putting all the points of into a matrix and plotting in 2D

- Here is a scatter plot to see the relationship between the two measures. A strong correlation with R = 0.903 can be seen on the log graph.
- While the citation count and PageRank show a strong correlation, many points stick out to the lower right on the graph, indicating that there are many papers with many citations but low PageRank.

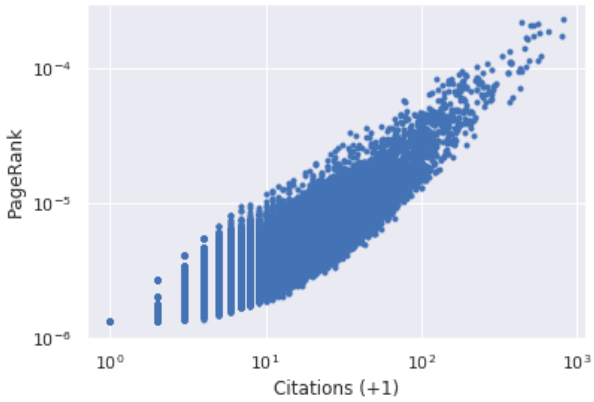

Figure . Putting all the points of into a matrix and plotting in 2D

- Sorting is based on citations where the condition of page rank is greater than 3*10 power -6

	title	year	venue	citation	pagerank
39704	Applications of the gröbner fan to gene networ...	2008	ACM Communications in Computer Algebra	34	0.000033
56008	Research on the Influence of Tunnel Wall on Ra...	2009	Proceedings of the 2009 WRI International Conf...	37	0.000031
110687	The New Zealand Digital Library Project	1996		43	0.000036
22079	Automatic Real-Time Moving Target Detection fr...	2006	Proceedings of the 2006 International Conferen...	43	0.000030
3048	Microsoft Windows 2000 70-220	2003		45	0.000030
45944	Estimating tree crown size with spatial inform...	2007	International Journal of Remote Sensing	46	0.000033
3781	IBM System I Security Guide for IBM I5/Os Vers...	2006		54	0.000037
31160	Modeling the Content of Adaptive Web-Based Sys...	2006	Proceedings of the First International Worksho...	59	0.000034
24696	Conditional Fault-Tolerant Cycle-Embedding of ...	2006	Proceedings of the Seventh International Confe...	61	0.000036
37804	New second-and fourth-order accurate numerical...	2007	International Journal of Computer Mathematics	63	0.000035

Figure . Naive Bayes Conditional Probability

- Top 10 sorted values of titles from the citations are extracted

```
array(['Applications of the gröbner fan to gene network reconstruction (abstract only)',
       'Research on the Influence of Tunnel Wall on Radiation Field Caused by the Leaky Antenna',
       'The New Zealand Digital Library Project',
       'Automatic Real-Time Moving Target Detection from Infrared Video',
       'Microsoft Windows 2000 70-220',
       'Estimating tree crown size with spatial information of high resolution optical remotely sensed imagery',
       'IBM System I Security Guide for IBM I5/Os Version 5 Release 4',
       'Modeling the Content of Adaptive Web-Based System Using an Ontology',
       'Conditional Fault-Tolerant Cycle-Embedding of Crossed Cube',
       'New second-and fourth-order accurate numerical schemes for the nonlinear cubic Schrödinger equation'],
      dtype=object)
```

Figure . Naive Bayes Conditional Probability

E. TF-IDF

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TF-IDF can be calculated as follows:

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

Term Frequency (TF): TF factor for the term (t) in the document (d) determines how many times the term t occurs in document. It can be calculated as follows:

$$tf(t,d) = log(1 + freq(t,d))$$

Inverse Document Frequency (IDF): IDF is a measure of how "unique" the term is. Very common terms have a low idf; very rare terms have a high idf. It can be calculated as follows:

$$idf(t, D) = log(N/count(d \epsilon D : t \epsilon d))$$

Implementation Steps:

- Initially the program asks for the folder where we have all the documents of the dataset.
- Using this input the program loops over each document and then calculates word count and term frequency for each term in the current document.
- Later on inverse document frequency is calculated for the dataset.
- Using the values we have just calculated the program outputs TF-IDF scores for the terms of the documents in text format.

F. *AutoSuggestion*

This approach utilizes Lucene's Suggester implementation and supports all of the lookup implementations available in Lucene. The main features of this Suggester are: Lookup implementation pluggability, Term dictionary pluggability, Distributed support. Relevance score algorithm is one of the keys of autocomplete search. The classical way of calculating relevance score is based on term frequency (tf), inverse document frequency (idf) and field-length norm. In our approach the relevance of the word is calculated using Tf-idf.

$$RelevanceScore = Tf * Idf * norm$$

Term frequency means the more frequent a term appears, the more significance it is. Inverse document frequency implies that a word's relative weight is related to the inverse of its occurrences in all documents, making more common words less significant than the unusual one. Field-length norm describes that the longer the text, the less significant it is compared to the shorter ones.

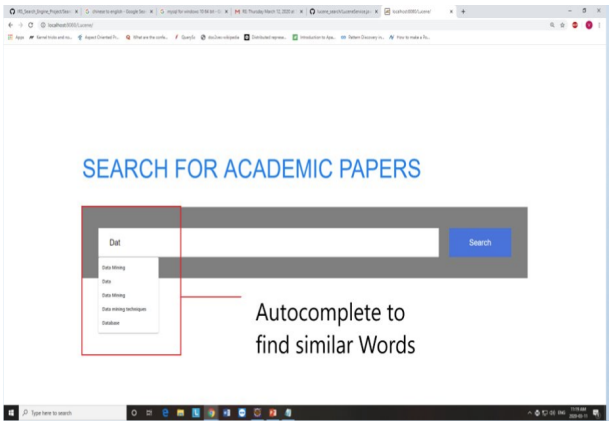

Figure . Putting all the points of into a matrix and plotting in 2D

G. *Word2vec*

Word2vec is an algorithm used to construct distributed representations of words representing word types; i.e., every given word in a vocabulary such as "get" or "grab" or "go", has its own word vector, and these vectors are stored in a lookup table or dictionary. This dictionary is also provided to Lucene Spell Checking component. Word Embedding is a mapping of words into vectors of real numbers using the neural network, probabilistic model, or dimension reduction on word co-occurrence matrix. Word2vec uses any of the two architectures namely Continuous Bag of words (CBOW) or Skip Gram to train the model. The CBOW model architecture tries to predict the current target word (the center word) based on the source context words (surrounding words). Skip-gram is one of the unsupervised learning techniques and predicts the context word for a given target word.

Implementation Steps:

- For implementing word2vec "gensim" Python package is used. Datasets: ICSE and VLDB - Sigmod
- NLTK and Word2vec are used together to find similar words representation or syntactic matching
- Word2vec takes a large corpus of text as input and produces a vector space of several hundred dimensions with each unique word in the corpus being assigned a corresponding vector in space
- Several hyper parameters including size=150, window=10, min_word_count=3, context_size=7, seed=1, workers=10, epochs=100 are considered while training the model
- Once the model is trained using gensim, the model is saved in vector.bin which is compatible with Lucene using the following:
  w2v_model.wv.save_word2vec_format
  ("path/to/w2v_model.bin", binary=True)

```
thrones2vec.wv.most_similar("knowledge")
```

```
[('Knowledge', 0.5410762429237366),
 ('shark', 0.3559122681617737),
 ('tacit', 0.33473658561706543),
 ('replaying', 0.3311447501182556),
 ('SHARK', 0.3134819567203522),
 ('consolidating', 0.3125236928462982),
 ('adversarial', 0.3042372167110443),
 ('reusing', 0.29766809940338135),
 ('Controlling', 0.29374587535858154),
 ('Genome', 0.2871089577674866)]
```

Figure . Putting all the points of into a matrix and plotting in 2D

H. *DOC2vec*

Doc2vec (aka paragraph2vec, aka phrase embedding) modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as words, paragraphs or whole documents. Since the Doc2Vec class extends gensim's original Word2Vec class, many of the

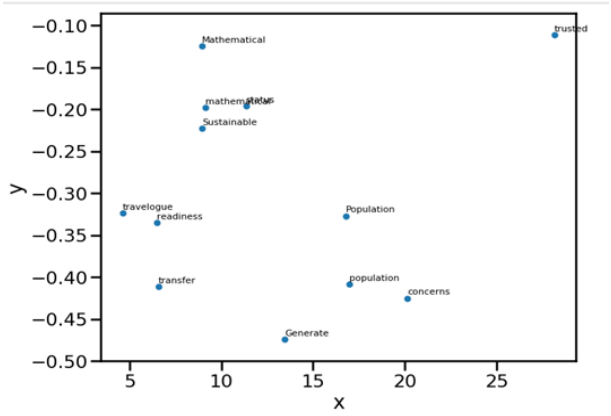

Figure . Putting all the points of into a matrix and plotting in 2D

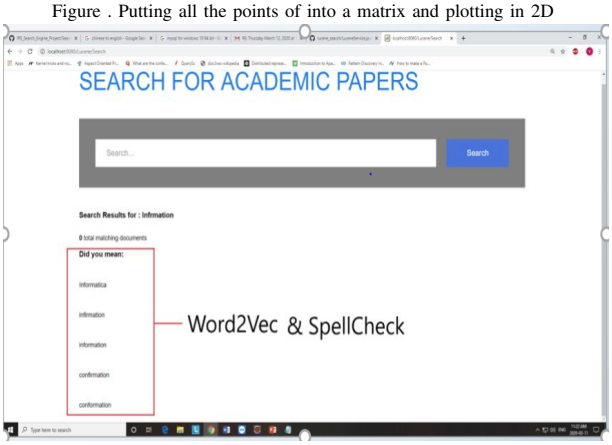

Figure . Putting all the points of into a matrix and plotting in 2D

usage patterns are similar, we can easily adjust the dimension of the representation, the size of the sliding window, the number of workers, or almost any other parameter that you can change with the Word2Vec model.

The only exception to this law are the parameters corresponding to the model's training process. In the word2vec architecture, the two algorithm names are "continuous bag of words" (cbow) and "skip-gram" (sg); in the doc2vec architecture, the corresponding algorithms are "distributed memory" (dm) and "distributed bag of words" (dbow).

1) Implementation Steps:

- **DBOW**
  DBOW is the Doc2Vec model analogous to Skip-gram model in Word2Vec. The paragraph vectors are obtained by training a neural network on the task of predicting a probability distribution of words in a paragraph given a randomly-sampled word from the paragraph. Training a Doc2Vec model is rather straight forward in Gensim, we initialize the model and train for 30 epochs: We set the minimum word count to 2 in order to discard words with very few occurrences.
- **Distributed Memory with Averaging**
  Distributed Memory (DM) acts as a memory that remembers what is missing from the current context or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. We again instantiate a Doc2Vec model with a vector size with 300 words and iterating over the training corpus 30 times

I. Comparison between word2vec and Doc2vec

Doc2vec performs better in finding the most similar words from the dataset and is an enhancement of Word2vec. We can see results for the word "clustering" below:

```
thrones2vec.wv.most_similar("clustering")
```

```
[('Clustering', 0.4644298851490021),
 ('partitional', 0.43912678956985474),
 ('categorical', 0.43032801151275635),
 ('wavecluster', 0.42839357256889343),
 ('clues', 0.37007856369018555),
 ('centroid', 0.3656623363494873),
 ('affinity', 0.3649447560310364),
 ('dominant', 0.3617939352989197),
 ('Categorical', 0.35365739464759827),
 ('relaxation', 0.35197728872299194)]
```

Fig 21. Word2vec Results

```
model.wv.most_similar('clustering')
```

```
[('k-means', 0.8385525941848755),
 ('text', 0.826462984085083),
 ('filtering', 0.7791950702667236),
 ('categorization', 0.7708678841590881),
 ('classification', 0.7617677450180054),
 ('similarity', 0.7384018301963806),
 ('watermarking', 0.731056272983551),
 ('weighted', 0.73093181848526),
 ('duplicate', 0.729840874671936),
 ('naïve', 0.7297391295433044)]
```

Fig 21. Doc2vec Results

VI. Classification and Clustering using Machine Learning Algorithms

A. Feature Selection

Reduction of feature dimensionality is of considerable importance in machine learning because it can reduce the computational complexity and it may improve the performance of the induction algorithm. Moreover, reducing the number of features results in better understanding and interpretation of the data. Feature selection reduces the number of original features by selecting a subset of them that still retains sufficient information for obtaining a good performance result.

*1) **Tf-Idf-Vectorizer**:*

*2) **Chi Squared**:*

B. Classification

*1) **OnevsRest Classifier**:* This strategy, also known as one-vs-all, is implemented in OneVsRest Classifier. The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice.

*2) **Naive Bayes**:* Naïve Bayes is a conditional probability model, given a problem instance to be classified, represented by a vector x = (x1, . . . , xn) representing some n features (independent variables), it assigns to this instance probabilities for each of K possible outcomes or classes.

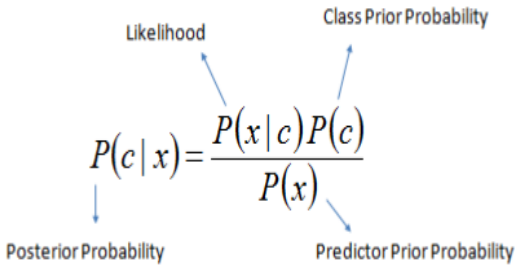

Figure . Naive Bayes Conditional Probability

This classifier takes into consideration the probability of a datapoint (x) belonging to a particular class (c). This probability value is calculated for all the classes and the datapoint then becomes a member of the class that has the highest probability for that particular datapoint.

Multinomial Naive Bayes Classifier: The Multinomial Naive Bayes is a variant of the Naïve Bayes classifier that is used for text classification. This classification algorithm works upon the probability of a given word in a document. The formulae can be given as:

P(word=w/document=d1) = Count (w in d1)/Count (total words in d1).

Here we are trying to find out the probability of a word w in a given document d1. This can be calculated by counting the word w in the document d1 divided by the total number of words in the document d1. Multinomial Naive Bayes gives the best accuracy when there are a lot of features and the data of these features is in the form of frequencies.

*3) **Logistic Regression**:* Logistic Regression uses the logistic function for classification. This function is also called as the sigmoid function which is in the form of an S shape. This curve can take any real value and provides the result between 0 and 1. The sigmoid function can be given by:

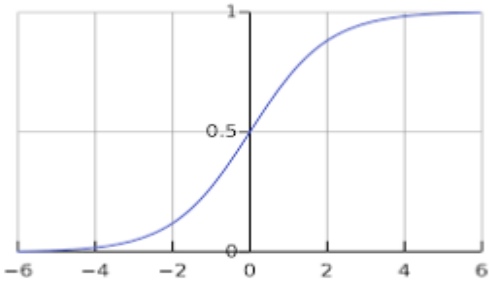

Figure . Naive Bayes Conditional Probability

Here, e represents the exponential function and i represents the input for which we want the output. This machine learning algorithm uses a equation just as used in linear regression. To find out the output say, y, a combination of input values and weighs is used. In case of linear regression, we get a continuous output value but in case of logistic regression, we get an output class. The equation for logistic regression is given by:

y = power(e,(a0 + a1*x)) / (1 + power(e(a0 + a1*x)))

*4) **Support Vector Machine**:* Support vector machine constructs hyperplanes in infinite-dimensional spaces to achieve classification. Intuitively,a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general larger the margin the lower the generalization error of the classifier.

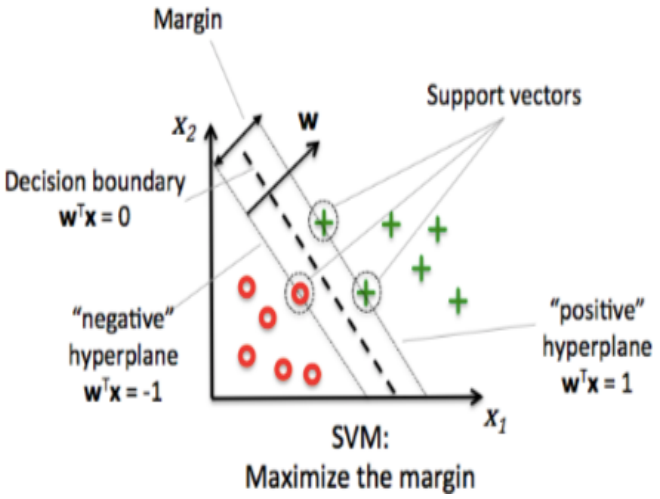

Fig 21. Support Vector Machine

Kernel in the SVM is responsible for transforming the input data into the required format. It is a a method of using a linear classifier to solve a non-linear problem. The kernel function is applied on each data instance to map the original non-linear observations into a higher dimensional space in which they become separable. Some of the kernels used in SVM are linear, polynomial and radial basis function (RBF),

sigmoid etc.

SVM minimizes the misclassification errors by maximizing the margin. We have used RBF which non-linearly separate the variables. For distance metric, squared Euclidean distance is used.

*5) **Cross Validation using StratifiedShuffleSplit**:* The ShuffleSplit iterator will generate a user defined number of independent train / test dataset splits. Samples are first shuffled and then split into a pair of train and test sets.

It is possible to control the randomness for reproducibility of the results by explicitly seeding the random state pseudo random number generator. ShuffleSplit is thus a good alternative to KFold cross validation that allows a finer control on the number of iterations and the proportion of samples on each side of the train / test split. StratifiedShuffleSplit is a variation of ShuffleSplit, which returns stratified splits, i.e which creates splits by preserving the same percentage for each target class as in the complete set.

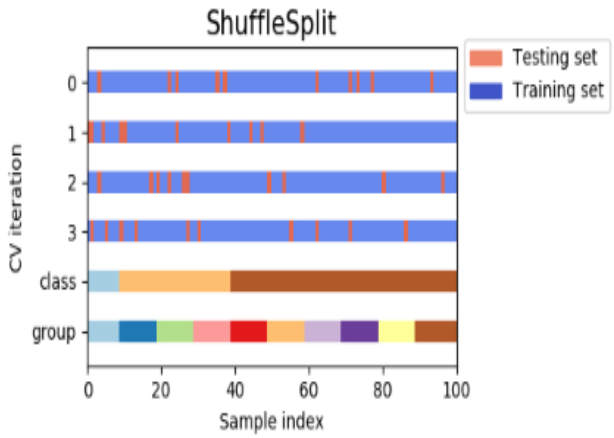

Fig 21. Support Vector Machine

Here is a visualization of the cross-validation behavior.

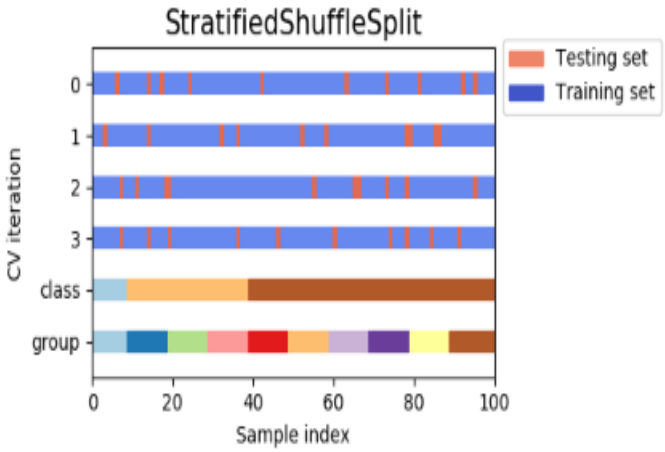

Fig 21. Support Vector Machine

C. *Clustering*

*1) **K- Means Clustering**:* K means is a form of unsupervised learning learning, also the main aim of k means is to find the data groups which are present in the given datasets which is given by the variable "k" in k-means , also we use value as k=3 in the given datasets for the operation because we plot 3 clusters for 3 datasets namely, sigmod, icse and vldb.

Also, we make use of the euclidean distance in the K means upon all the datasets because squared deviation summation from a reference point which in this case we consider as centroid is the sum of squared euclidean distance of pairwise values under division of number of points which provides better results and hence we consider euclidean distance over any other distances

*2) **Dimensionality Reduction using PCA**:* PCA (Principal Component Analysis) transforms the dataset from the original coordinate system to new coordinate system where new coordinates are chosen by data itself The general method to obtain PCA is given as follows:
Step1: Compute the mean
Step2: Compute the covariance matrix
Step3: Find the eigen values from largest to the smallest
Step4: We determine the top 'n' eigen values
Step5: Transform the data into the new space created by "n" eigen values which becomes the eigen vector

VII. Evaluation Metrics

A. *Precision*

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. Precision = TP/TP+FP

B. *Recall (Sensitivity)*

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? Recall = TP/TP+FN

C. *F1 score*

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. F1 Score = 2*(Recall * Precision) / (Recall + Precision)

VIII. Results

IX. Conclusion and Future Work

A Search Engine is built for searching academic conference papers. Basic functionalities of Search Engine like Indexing and Searching have been implemented and enhanced with improvements that include fast searching and query processing. Moreover, features added include Highlighting, Spellchecking, Autocomplete, word2vec, doc2vec, Page Ranking, TF-IDF, Feature Selection, Classification and Clustering.A basic GUI is built for the Search Engine to display the results.Further, it can be expanded to build a proper website with an interactive GUI and work with larger datasets in other domains.

References

[1] http://jlu.myweb.cs.uwindsor.ca/538/
[2] https://introcs.cs.princeton.edu/java/16pagerank/
[3] https://medium.freecodecamp.org/
[4] https://rare-technologies.com/
[5] https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1
[6] https://cwiki.apache.org/confluence/display/LUCENE/PageRank