EasySearch: Semantic Aware Search for Local Machine

Enrol. No. (s) - 16803007, 16803009, 16803015

Name of Student (s) - Sugandha Singhania, Urvi Agarwal, Nayan Garg

Name of supervisor - Dr. Shikha Jain

May-2020

Submitted in partial fulfillment of the Degree of

5 Year Dual Degree Programme B. Tech

In

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

DECLARATION

I/We hereby declare that this submission is my/our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signatures:	……	…….	…….
Name:	Urvi Agarwal	Nayan Garg	Sugandha Singhania
Enrollment No(s):	16803009	16803015	16803007

Date:

Place:

CERTIFICATE

This is to certify that the work titled **"EasySearch"** submitted by **Sugandha Singhania, Urvi Agarwal** and **Nayan Garg** in partial fulfillment for the award of degree of 5 Year Dual Degree Programme of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor: ……

Name of Supervisor: **Dr. Shikha Jain**

Designation: ASSISTANT PROFESSOR (SENIOR GRADE)

Date:

ACKNOWLEDGEMENT

We would like to extend a heartfelt thanks to our college Jaypee Institute of Information Technology for giving us an opportunity to work on the project "**EasySearch**".

The successful completion of this project would not have been able without the support of our Supervisor Dr. Shikha Jain. In every phase of this project, her support and guidance were very valuable for us.

We also want to thank our friends for supporting us and giving their suggestions during the development of this project.

Signatures:	……	…….	…….
Name:	Urvi Agarwal	Nayan Garg	Sugandha Singhania
Enrollment No(s):	16803009	16803015	16803007
Date:			

SUMMARY

"EasySearch" is a push to straightforwardness and advances the procedure of local search, for example off the web search on a person's PC or laptop. Specially aimed at research scientists who have a humongous number of research papers saved offline; a great deal of manual work is required to fetch documents holding words or phrases that are related to a certain topic. A simple example could be papers related to machine learning. While scanning for this expression, it would be of colossal assistance if papers identified with subjects like Neural Nets, Artificial Intelligence and so forth likewise turn up. This is the distinctive feature of our project, apart from exact word matching our model is robust in displaying semantically similar words to a user's query.

The aim- "a semantic search" is kept in mind from the very beginning as the first process of this task itself is semantically aware. We utilize a parameter less method for keyphrase extraction by constructing a graph of text that captures the contextual relation between words. A word scoring strategy is likewise utilized dependent on the association between themes or concepts. Once keywords and keyphrases are extracted from every document, a connection between all the extracted words is made with the help of word embeddings. The entries are fed into a database of words. A user can look for a specific subject by entering a query, the input query is pre-processed and the results are fetched from the database. The corresponding term and the terms that are closest to it in the database are displayed as the output along with metadata of related documents.

Signature: …….

Name: Sugandha Singhania (16803007)

Signature: ……

Name: Urvi Agarwal (16803009)

Signature: ……

Name: Nayan Garg (16803015)

Signature of Supervisor: ……

Name of Supervisor: **Dr. Shikha Jain**

Date:

LIST OF FIGURES

LIST OF TABLES & GRAPHS

LIST OF SYMBOLS & ACRONYMS

Symbol or Acronym	Full Form
NLP	Natural Language Processing
sCAKE	Semantic Connectivity Aware Keyword Extraction
GloVe	Global Vectors for Word Representation
NLTK	Natural Language Toolkit
BERT	Bidirectional Encoder Representations from Transformers
CAG	Context Aware Graph
ACM	Association for Computing Machinery
RAKE	Rapid Automatic Keyword Extraction
KB	Knowledge Base
AUC	Area Under Curve

1. INTRODUCTION

1.1 GENERAL INTRODUCTION

Browsers and search engines have become progressively well-known on the grounds that they offer unrivalled precision and additional highlights. Unfortunately, there are very rare Search Engines for your local machine, say PC or laptop which could give a similar approach for hassle-free search of the gigantic measure of information present these days in the filesystem that is difficult and tedious to keep up with. Desktop search tools search within a user's own computer files as opposed to searching the Internet. These tools are intended to find information on the user's PC, including web browser history, email archives, text documents, sound files, images, and video. For scientists and researchers who work on a variety of domains and datasets, it is hard to maintain all the data present and significantly harder to search for a specific query. This project proposes the utilization of AI procedures such as Natural Language Processing to ease this problem faced by researchers all over the world. We intended to work in domains of natural language processing such as text classification and information extraction that enables our semantic pursuit and identifies informative documents. Throughout the years many desktop search engines came up like Docfetcher [21], Beagle++ [20], Locate32, Recoll, Spotlight, etc. But they focus mainly on searching among different file types and performing linguistic inquiry on text documents. To overcome this gap, we propose a local search engine, **EasySearch**, which will furnish users with priority results over selected document pool. This local search engine can search a document; it will first refine the query for better comprehension of problem and finding all possible solutions in accordance with the requests of users.

1.2 PROBLEM STATEMENT

Researchers need to pursue a ton of papers, surveys and journals for writing one quality paper. Each time a research scientist needs to find such articles, he/she needs to explore the length and breadth of the internet and download them. It is typical for such insightful articles to not have any appropriate titles once downloaded. Most file search engines available in the operating systems are not capable of searching within documents and regardless of whether they do, they are simply performing precise string matching which is a long way from the effective and accurate semantic search implemented by online search engines like Google. When the researcher needs these documents again, he/she would either have to open each document and check manually, or find them again on the internet. This results in not only a waste of effort but also a tremendous chaos in

the downloads organizer. Consequently, it will be of great help to have a search engine to search semantically within documents in the local file system.

The aim is to develop a tool for the research community which can be used to search for the required documents in an efficient and accurate manner.

1.3 NOVELTY OF THE PROBLEM

We have done a careful search of the tools accessible for search on a user's Local Drive. Despite the presence of many state-of-art tools, none of them is proficient in semantic search. Thus, this project would be first of its kind. A critical advance in the work-flow was keyphrase extraction, for which in the wake of contemplating different techniques and algorithms we came across SCake[9] which is uses a semantically aware algorithm and has results better than other well-known algorithms for same the purpose. We are proposing an extension to the algorithm hence maintaining the originality of the project. This project is being designed by keeping in mind the needs of a research scientist and would be highly beneficial to them. It would cater to their requirement of searching from a vast variety of documents on their Desktop. As opposed to search on the web which would display infinite search results, user would be able to search among the documents he/she wishes to.

1.4 EMPIRICAL STUDY

An intensive study of sub fields of Natural language processing keyphrase extraction, topic modelling, word embeddings and so-forth is required while building an undertaking based on it. Broad investigation has been done in these sub fields as these are mainstream popular research topics. We have attempted to alter and consolidate the best highlights of every one of these procedures and calculations and incorporated these into our task. Desktop search tools search within a user's own computer files rather than looking through the Internet. Tropes Zoom was one such search engine which suggested that the user replace each identified word with a hypernym. In this manner, the pursuit isn't done legitimately on the words picked, however on the totality of their semantic reciprocals. But the Tropes Zoom software is no longer available. Although various Desktop Search Engines like Beagle++, Docfetcher are present out there, but the search is limited to word matching only. We broaden this limit by bringing semantic search into the project.

In this project, we have used a graph-based algorithm for keyword extraction. Further, Glove [16] is used for vector representation of the keywords and key phrases extracted and concluding the word pair affinity. Various sources [9], [36] have proved that graph-based algorithms work better than machine learning techniques for keyphrase extraction. Using the conclusions drawn from these papers and journals we resolved on using the same for our keyphrase extraction phase. For the word embeddings part, we studied classic models word2vec [15] as well as new state of the art models like BERT [18], lda2vec [17] and Glove [16]. Between all these Glove suited our requirements best. Thus, the final model was created after comparing substantial algorithms and techniques. Further, for accessing the words once going through Glove, we explored both a graph-based approach and an approach of maintaining a database. Conclusively, we chose the database approach as it provides faster access to words, faster updating and is fault tolerant as compared to a knowledge graph.

1.5 BRIEF DESCRIPTION OF THE SOLUTION APPROACH

Like any other search engine, EasySearch will require the user to enter a query relating to the records he needs to search. Thus, he would get a rundown of the documents having keywords and topics with most similarity to the query, ranked in order of relevance. We have done this work in phases: Keyphrase extraction, creating word embeddings and finding closeness of the words, feeding the words in the database and query processing and matching. Keyphrase extraction is done using a graph-based algorithm and once the key phrases are extracted, the documents along with respective phrases are fed into the pre-trained glove model. Similarity metric Cosine distance is used for connecting the words semantically. MySQL is used for database creation, which also stores the metadata for each word. On entering a simple query, in the form of a word or a phrase, it is first pre-processed and then top key phrases related to that word are displayed along with corresponding metadata which stores information like name of the file.

2. LITERATURE SURVEY

2.1 SUMMARY OF THE PAPERS STUDIED

S. No.	Paper Name	Publication and Year	Brief Summary	Advantages	Gaps
1.	**sCAKE: Semantic Connectivity Aware Keyword Extraction [9]**	Information Sciences, 2018	sCAKE is a parameter-less keyphrase extraction strategy that utilizes the semantic availability of words in a record for its potential benefit. It uses a context aware graph (CAG) and scoring strategy and results in a novel parameter less keyword/phrase extraction technique.	It is a parameter-less keyphrase extraction strategy. It additionally follows two significant ideas, pragmatics and entailment. The window size is taken as two back to back sentences which guarantees setting stream. The calculation likewise takes less time when contrasted with other techniques for graph development. The semantic network of words is considered.	The scoring method suggested in this paper is purely based on empirical results by the author only. Words are considered as an individual entity by the proposed technique. Only nouns and adjectives are considered for node creation in the CAG, missing on verbs like "learning".
2.	**A Position-Biased PageRank Algorithm for Keyphrase Extraction [8]**	AAAI, 2017	This keyphrase extraction technique is an unsupervised graph-based approach. It uses a biased PageRank algorithm and feeds in the information from all positions of words occurrences into it. Frequency of a word as well	The improvement shown in results by this technique is as high as 26.4%. This model presents better results than strong baselines.	The window size directly affects the results of the algorithm and this deciding parameter is left upon the user to decide. Also, the semantic aspect is left out by the proposed

			as its relative positioning is considered in this technique.		technique.
3.	**A Novel Extension for Automatic Keyword Extraction [5]**	International Journal of Advanced Research in Computer Science and Software Engineering, 2016	Automatic keyword extraction empowers us to distinguish a little arrangement of words, from a literary archive with the assistance of which the importance of the report can be depicted.	In this paper, a popular keyword extraction technique, RAKE is studied in detail and a novel extension of RAKE is proposed so as to improve the performance of RAKE further.	If you don't have a comprehensive list of stop words the phrases can get quite long and less useful.
4.	**Keyword and keyphrase extraction techniques: a literature review [1]**	International Journal of Computer Applications, 2015	Keywords and keyphrases are set of delegate expressions of a document that gives a high-level specification of the theme for intrigued users.	This paper helped in better comprehension of different keyword/keyphrase extraction technique present till 2015. This provided a broad overview of the topic.	This was a literature review (survey paper) and hence didn't accommodated any gaps.
5.	**Main Core Retention on Graph-of-Words for Single-Document Keyword Extraction [7]**	Advances in Information Retrieval. Springer, 2015	The definition of k-core on the graph-of-word representation of text for extraction of a single-document keyword was introduced in this paper, keeping only the nodes from the main core as symbolic words.	This approach takes better account of the similarity between keywords and variation in the number of extracted keywords by choosing more coherent node subsets than with current centrality-based graphic approaches. It shows statistically significant improvements in the precision / recall curve F1-score and AUC compared to	The approach achieves a high degree of accuracy but this is done at the expense of precision, since it tries to find just a few cores with lots of vertices that lower the accuracy but ensure minimum recall. This approach also appears to overestimate keyword counts.

				baseline results	
6.	**DegExt—A language-independent graph-based keyphrase extractor [6]**	Advances in Intelligent Web Mastering, Springer, 2011	DegExt is a graph-based language independent keyphrase extractor, which extends the keyword extraction method.	Degext has advantage over TextRank and GenEx in terms of precision and AUC for a total of fifteen keyphrases or more.Moreover, DegExt outperforms both GenEx and TextRank in terms of implementation simplicity and computational complexity.	This better performance th at DegExt shows is at the cost of decrease in recall and F-measure.
7.	**Automatic keyword extraction from individual documents [4]**	Text mining: applications and theory, 2011	RAKE is based on the observation that keywords frequently contain multiple words but rarely contain standard punctuation or stop words, such as the function words and, the, & of, or other words with minimal lexical meaning.	RAKE takes a straightforward arrangement of information parameters and consequently removes keywords in a solitary pass, making it reasonable for a wide scope of documents and collections.	It doesn't have the setting of outside word utilization so can't tell as effectively what is generally essential to this particular record.
8.	**Single Document Keyphrase Extraction Using Neighborhood Knowledge [3]**	AAAI, 2008	This paper proposes to use a small number of nearest neighbour documents to provide more knowledge to improve single document keyphrase extraction.	Experimental results show the viability and power of the proposed approach. Exploits global data from the neighbouring documents and local data from the predetermined document.	The changing of number of neighbouring documents pool and window-size affects the result tremendously. The proposed approach has higher computational multifaceted nature than the standard methodology since it

					includes more records.
9.	**Textrank: Bringing order into text [2]**	EMNLP'04, 2004	This weighted and unsupervised graph-based method takes advantage of PageRank scoring model and picks out top n-words as keywords of the given document.	Keyphrases are built in the post processing stage of this approach. Co-occurrences of potential keywords in the same window results in a keyphrase.	The weight of providing carefully tuned parameters to the graph construction method as well as the word scoring model is left on the user's shoulder. The semantic connectivity is ignored and the algorithm only acknowledges co-occurrence relationship among potential keywords.
10.	**Tf-idf: Term Frequency Inverse Document Frequency [14]**	"Data Mining": Mining of Massive Datasets, 2011	Term Frequency Inverse Document Frequency is one of the feature extraction techniques in NLP which deals with providing weights to each word indicating how important that word is with respect to overall documents.	It is easy to compute and we can have some basic metrics to extract the most descriptive terms in a document. Good for finding similarity between two documents.	A major issue with tf-idf method is that it is corpus dependent, which in turn restricts its applicability to dynamic collections.
11.	**Bert: Pre-training of deep bidirectional transformers for language understanding [18]**	arXiv preprint, 2018	BERT is designed to pretrain profound bidirectional representations from unlabeled text by jointly conditioning in all layers on both the left and right context.	BERT makes a significant contribution by further generalizing unsupervised pre-training to deep bidirectional architectures, allowing the same pre-trained model to effectively	BERT generates contextually dependent embeddings and hence word-level similarity comparisons become inappropriate.

				solve a wide variety of NLP tasks.	
12.	**Deep contextualized word representations [19]**	arXiv preprint, 2018	This paper introduces a new type of deep contextualized word representation that models both: 1- complex attributes of word use (e.g., grammar and semantics) 2- how these utilizations shift across linguistic contexts.	Elmo uses biLM layers which efficiently encode various kinds of syntactic and semantic data about words in setting, and that using all layers improves overall task performance.	We required a model that was used to train the vectors independent of context and since it is context dependent, it was not suitable for our task.
13.	**Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec [17]**	arXiv preprint, 2016	LDA2vec is a topic model that extends the word2vec model with thematic and text vectors and integrates concepts from both word embedding and subject models, which is inspired by the Latent Dirichlet Allocation (LDA).	This is easy to integrate into existing automated classification systems and enables scientists to use unsupervised text representations while simultaneously studying word vectors and their linear relationships.	Training of lda2vec is a computationally intensive process.
14.	**Glove: Global vectors for word representation [16]**	Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014	Glove is a global log bilinear regression model incorporating the benefits of the two main model families in the literature: global matrix factorization and window approaches for the local context.	By considering the relationships between word pair rather than word and word, Glove adds some more practical meaning into word vectors. For extremely repeated word pairs it gives lower weight to avoid pointless stop words.	The memory requirement of the model is high because it is trained on the co-occurrence matrix of words. Tuning of hyper parameters leads to reconstruction of the matrix, which is a

					time-consuming process.
15.	**Distributed representations of words and phrases and their compositionality. [15]**	Advances in neural information processing systems, 2013	Several extensions are suggested which boost both the quality of the vectors and the pace of training by subsampling the frequent terms. Authors often define a basic alternative to the SoftMax hierarchy, called negative sampling.	The concept is very intuitive, translating the unlabeled raw corpus into labeled data (by mapping the target word to its background term) and learning to represent words in a process of classification.	The theoretical backing is weak and the sub-linear relationships are not explicitly defined.
16.	**An introduction to latent semantic analysis [10]**	Discourse processes, Taylor & Francis, 1998	Latent Semantic Analysis (LSA) analyses documents to find the underlying meaning or concepts of those documents. If each word only meant one concept, and each concept was only described by one word, then LSA would be easy since there is a simple mapping from words to concepts.	Easy to implement, understand and use. There are many practical and scalable implementations available like mahout (java), gensim (python), scipy (svd python).	The model is not humanly readable. Debug/evaluation is possible through finding similar words for each word in the latent space though. But otherwise not easy to interpret like LDA.
17.	**Topic modelling for qualitative studies [13]**	Journal of Information Science, SAGE, 2017	This paper proposes a new quality metric, tf-idf coherence, that reflects human judgement more accurately than regular coherence, and an interval semi-supervised approach (ISLDA) where certain predefined sets of	Their experiments show that ISLDA is better for topic modelling than LDA in terms of tf-idf coherence, number of topics identified to predefined keywords and topic stability.	The paper does not answer the question how to best extend the tf-idf coherence metric (which at present deals with topics) so that it can serve as a good measure for the

			keywords are restricted to specific intervals of topic assignments.		overall quality of a specific topic model or run.
18.	**A heuristic approach to determine an appropriate number of topics in topic modelling [12]**	BMC bioinformatics, 2015	Choosing the best number of subjects, as well as other latent variable methodologies, is a problem in subject modeling. The paper offers the same solution.	The proposed RPC (rate of perplexity change) dependent approach is seen to pick the best number of topics in three numerical studies with vastly different types of data, and for very different size datasets.	To substantiate the proposed method's theoretical claims, further investigation is required. Also, for the establishment of generalization across dataset characteristics additional work is required.
19.	**Latent Dirichlet Allocation [11]**	Journal of Machine Learning Research 3, 2003	Latent Dirichlet allocation (LDA), is a generative probabilistic model for assortments of discrete information, for example, text corpora. LDA is a three-level progressive Bayesian model, wherein everything of an assortment is displayed as a limited blend over a fundamental arrangement of topics.	As a probabilistic module, LDA can be promptly inserted in an increasingly complex model a property that isn't controlled by LSA. Some different points of interest of generative models, for example, LDA incorporate their particularity and their extensibility. .	The disadvantages are that it is hard to know when LDA is working --- subjects are soft-clusters so there is no target metric to say "this is the best choice" of hyperparamete rs.
20.	**Applications of JNI Technology in Desktop Search Tool Docfetcher [21]**	Computer Technology and Development, 2013	Docfetcher is an open source, cross platform portable Desktop search engine developed using Apache's Lucene and supports major formats like pdf,	Main features of Docfetcher which come in handy are its portability, zero cost, efficient search for exact words and its indexing search which enables	Docfetcher enables only fuzzy search which gives results like 'roam' when the user searches for 'foam'. We

			word, epub etc.	users to search among documents he wants instead of indexing the entire hard drive.	aim to achieve higher than Docfetcher by including semantic search in our project.
21.	**Beagle++: Semantically enhanced searching and ranking on the desktop [20]**	European Semantic Web Conference, Springer, 2006	Beagle++ uses a 3-layer architecture for search on a Desktop. It makes use of metadata for usage context.	The advantage lies in the usage of metadata of files and in its ability to rank the results which enables the user to quickly find the most applicable resources.	Beagle++ considers semantics related to metadata only, e.g. how recently a file was opened or edited. This does not solve our problem statement.
22.	**Clustering in Weighted Networks [24]**	Social networks, 2009 - Elsevier	A generalization of global clustering coefficient that holds the data encoded in the weights of ties.	Unlike what is normally done with the standard clustering coefficient, this summed up measure takes into consideration the weights of ties in weighted networks utilizing various parameters.	The weighted local clustering coefficient is inevitably one-sided by the way that it assembles unequivocally on the neighbourhood binary coefficient.
23.	**Stemming and lemmatizatio n: A comparison of retrieval Performance [25]**	Proceedings of SCEI Seoul Conferences, 10-11 Apr 2014	Stemming and lemmatization techniques of text processing are compared with each other and a baseline method (without language processing).	The results show that language processing techniques improve the relevancy of document retrieval compared to the baseline algorithm. Lemmatization on the other hand, yielded more relevant results when compared to stemming.	The principle downside is the test assortment. During the assessment, it was discovered that the majority of the inquiries were not appropriate to be utilized for a language model as they don't contain things that require stemming or lemmatization.

24.	**Specializing Word Embeddings for Similarity or Relatedness [26]**	In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.	They exhibit the upside of practicing embeddings for the undertakings of genuine similarity and relatedness looking at two retrofitting strategies and a joint learning approach.	This work shows the advantage of specializing semantic spaces for either similarity or relatedness. Specializing for similarity is achieved by learning from both a corpus and a thesaurus.	These specializations are restricted to the equivalent words presented. As on account of MyThes venture 80,000 equivalent words are presented for coordinating the embeddings towards similarity.
25.	**Collocations [27]**	Handbook of Natural Language Processing, 2000	Collocations are characterized with the assistance of models, just as techniques for their extraction and order. Their utilization for word disambiguation, content age, and machine interpretation are likewise examined.	An in-depth study about collocations including methods of collocation extraction from text corpora, its uses and translation spanning over different languages is presented.	A drawback to this chapter is that it does not talk about the various different types of collocations.
26.	**Short Text Similarity with Word Embeddings [22]**	In Proceedings of the 24th ACM international on conference on information and knowledge management, 2015	This paper proposes to go from word-level to text-level semantics by consolidating experiences from techniques dependent on external resources of semantic information with word embeddings.	An arbitrary number of word embedding sets can be Incorporated and various kinds of meta-highlights from the examination of the word vectors for short content sets, and from the vector methods for their separate word embeddings are determined.	An apparent confinement of computing meta-features in the way they propose, i.e., from mean word vectors and word alignments, is that the order of words isn't considered.
27.	**Using WordNet as a Knowledge**	In Proceedings of the AICS Conference,	The utilization of WordNet as an information base	A noteworthy favourable position of	An essential prerequisite to this application

	Base for Measuring Semantic Similarity between Words [23]	1994	in a data retrieval task is proposed. A semantic similarity measure is presented which can be utilized as a choice to pattern matching in the examination process.	building KB is the way that the outcomes will fill in as exhaustive beginning stages in getting HCGs that contain all relevant concepts in an information domain.	is the accepted presence of a sense disambiguator which can automatically tag words from the WSJ articles with their appropriate KB meanings.
28.	**A Survey of Text Similarity Approaches [28]**	International Journal of Computer Applications,2013	This is a survey paper which describes various text similarity measures and algorithms on different levels and uses the SimMetrics package to implement them.	A variety of String Based algorithms (character-based and term- based), Corpus-Based and knowledge-Based algorithms are discussed. Samples of combinations between similarity algorithms are also introduced.	This paper does not present any comparative results for the algorithms mentioned.
29.	**Dynamic Hashing: Adaptive Metadata Management for Petabyte-scale File Systems [31]**	23rd IEEE/14th NASA Goddard Conference on Mass Storage System and Technologies. 2006	Dynamic Hashing strategy is proposed in this paper to deal effectively with growing metadata size for petabyte-scale file systems as metadata performance and accuracy affect overall system performance and efficiency.	DH can change working load dynamically. Often, with MDS (database server) being added or deleted, DH will reduce the database change and do so in parallel. Bottlenecks are easily eliminated by DH. Requests circulated fewer. It offers high-performance metadata storage and scalability for incredibly large file systems.	The dynamic process incurs a little overhead is the main drawback of this approach.
30.	**Metadata Impact on Research**	International Conference on Theory and	The paper proposes a strategy called ESA to find similar	Considering that some metadata forms clearly	The suggested approach is more complex

	Paper Similarity [32]	Practice of Digital Libraries. Springer, Berlin, Heidelberg, 2010	documents on the basis of metadata.	display promise, it seems promising to find methods for learning to classify papers randomly based on a mixture of abstract knowledge and other functions.	in numerical terms. The findings obtained on experiments are failing to significantly change abstract. The relative success of the ESA methods depends strongly upon the scale of the test set.
31.	**Knowledge graph refinement: A survey of approaches and evaluation methods [35]**	Semantic web 8.3 (2017): 489-508	The generated knowledge graphs aim to bring about a successful trade-off between completeness and correctness. To further enhance the usefulness of such knowledge graphs, various methods of refinement have been proposed in this paper which attempt to infer and add missing knowledge to the graph, or identify erroneous pieces of information.	It enables the creation of methods for improving arbitrary graphs of knowledge, which can then be extended to enhance several graphs of knowledge. Other than fine-tuning the heuristics that build a information list, the effect of these common methods of optimization that can have an enormous influence on the overall results is studied.	This is a survey paper. Different techniques of refinement are discussed.
32.	**Domain-Specific Keyphrase Extraction [33]**	Wu, Yi-fang Brook, et al. Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005	This paper describes a Keyphrase Identification Program (KIP), which uses prior positive samples of identified domain keyphrases to apply weights to the potential	In this paper the KIP algorithm suggested focuses primarily on pairs of adjective nouns. It uses an upgraded version of the Brill's tagger. A glossary database that contains domain-	The first step of the KIP algorithm includes populating a glossary database. The database is needed to be populated manually. The

			keyphrases to extract database key phrases.	specific manual keyphrases and keywords is used to calculate the scores for noun phrases, which provide initial weights for the keywords and sub-phrases of a candidate keyphrase.	manual labour is the major drawback of this algorithm.
33.	**NetworkX [34]**	Los Alamos National Lab (LANL), Los Alamos, NM (United States), 2008	NetworkX is a Python language program intended to test and analyse networks and network algorithms. The kit provides data structures with parallel edges and self-loops to represent several forms of networks, or graphs like simple graphs, directed graphs, and graphs.	The nodes in NetworkX graphs can be any Python entity (hashable) and edges can contain arbitrary data; this versatility makes NetworkX perfect for networking across multiple science fields.	It is available only as a python module.
34.	**Learning Global Features for Coreference Resolution [30]**	arXiv preprint, 2016	The model suggested uses recurrent neural networks (RNNs) to learn directly from their description of latent, global representations of entity clusters. These representations are useful for pronominal name estimation and can be integrated into an end-to-end co-reference scheme that exceeds the cutting edge	A simple, state-of-the-art approach to integrate global knowledge into an end-to - end coreference method that eliminates the need to describe global characteristics and also allows for simple (greedy) inference.	Further improvement can be done on obtained recall values. The scope of improvement can be extended to the global training approach.

			technology without requiring further search.		
35.	**Machine Learning for Entity Coreference Resolution: A Retrospective Look at Two Decades of Research [29]**	Thirty-First AAAI Conference on Artificial Intelligence, 2017	This is a survey paper which discusses multiple machine learning based approaches for coreferencing. Different supervised, unsupervised and semi-supervised methods are studied at great length.	Impressive efficiency is exhibited by supervised simulations of co-reference entities. In addition, to boost the efficiency of cross-task joint models involving entity coreference as one of a number of similar tasks to be studied, such that hard / soft cross-task constraints are proposed.	Models of supervised entity coreference cannot be extended to the vast majority of low-resource languages worldwide for which coreference-annotated data are not readily accessible. There is more room for development in the area.
36.	**Keyword extraction: a review of methods and approaches [36]**	University of Rijeka, Department of Informatics, 2014	This paper provides a study of techniques and solutions to the retrieval of keyword functions. The paper also provides a systematic analysis of current work, in addition to the systematization of processes.	This paper is a thorough systematization of current keyword extraction approaches: evaluating similar research on supervised and unsupervised methods with a special emphasis on the graph-based methods. The paper describes the centrality factors most often used which are critical in graph-based approaches.	Proposal for further investigation of SBKE method on different text types and other languages and for text summarization has been made.
37.	**A Simple algorithm for identifying abbreviation definitions in**	Pacific Symposium on Biocomputing, 2003	This paper proposes a novel algorithm to identify abbreviations,	The proposed algorithm achieves higher precision and recall over known methods	The algorithm takes into consideration the abbreviations

	biomedical text [37]		especially in biomedical texts.	for large datasets and similar values for smaller datasets as compared to other algorithms. A big advantage is that it does not require training data.	which are enclosed in parentheses and hence misses out on short forms such as acronyms which are used without brackets.
38.	**Comparison of abbreviation recognition algorithms [38]**	2010 REU Program.	A comparative study of abbreviation identifying algorithms is presented with the help of this paper.	Algorithms of various domains such as alignment -based, rule-based, machine learning are compared.	This was a literature review (survey paper) and hence didn't accommodated any gaps.
39.	**Semantic matching in Search [39]**	*Foundations and Trends® in Information Retrieval*, 2014	This survey includes a comprehensive and thorough detail about some state of art machine learning approaches for query and document matching.	Different aspects of query matching are discussed with main focus on semantic matching. It also focuses on different aspects of the solutions of query document matching like form aspect, phrase aspect, word sense aspect, topic aspect, and structure aspect.	This is a survey paper discussing future scope in document and query matching niche.

2.2 INTEGRATED SUMMARY OF THE LITERATURE STUDIED

For the fulfillment of this project and to gain knowledge about various methods and algorithms that were to be implemented for it, we had performed extensive research in certain fields. We studied research papers from different domains ranging from already existing Desktop Search Engines, keyphrase/keyword extraction to metadata storage, query pre-processing, word embeddings and semantic knowledge graphs. As explained in the section above we did a critical analysis of all the papers studied identifying their gaps and advantages. Accordingly, we have integrated the best and robust techniques into the proposed project. The first stage was to identify keywords and keyphrases. There are two methods for automatic keyword generation, Keyword/Keyphrase assignment and Keyword/Keyphrase extraction. The assignment approach endeavours to bound the arrangement of potential keyphrases by a predefined jargon of words. The goal of the former approach is to locate a little arrangement of terms that portrays an individual text document. The later approach selects distinctive keywords/key-phrases from a document.

Further various word embedding techniques were studied ranging from the very basic word2vec to latest ones like Glove and BERT. Word embedding is the aggregate name for a set of language modelling and feature learning techniques in natural language processing where words or expressions from the jargon are mapped to vectors of genuine numbers. Hybrid models which inherit features from both word embeddings such as lda2vec and other techniques related to the project which are of prominence in the NLP such as LDA, LSI and tf-idf were also explored.

We also studied papers for coreference resolution, stemming and lemmatization, abbreviation identification and collocations to make improvements in the keyword extraction algorithm implemented. WordNet and text similarity were studied to gain insights about useful methods for finding out similarity between words once they are represented in vector form. We have also studied various survey papers like comparative study between graph-based and machine learning techniques, survey on stemming and lemmatization and which is better, survey of hashing techniques and others. NetworkX is the python library we used to construct the graph model and also studied the corresponding paper. To identify various methods that could help in query reformulation, papers based on semantic matching in search and those related to query matching were thoroughly read. The identified methods and their shortcomings are tried to overcome in this project by using additional features in an integrated model for semantic word search.

3. REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 OVERALL DESCRIPTION OF THE PROJECT

The major phases of the project can be divided into four parts, namely Document Understanding, Database Creation, Query Understanding, and Matching and Ranking. These categories are described as follows:

a. Document Understanding

As the name suggests, document understanding is the process of exploring and gaining insights about the document which will lay the foundation for further work of the project. We evaluated various ways to choose one which would give the best information about a document, such as topic modelling techniques like LDA, LSI etc., tf-idf for word-document frequency, and keyword extraction methods based on machine learning and graph-based algorithms. After studying each technique, we concluded that graph-based keyword/keyphrase extraction will be best suited for this project.

In a brief description, keyword extraction can be explained as a task which automatically identifies the terms that best describe the core concept/theme of a document. Several such keyword extraction methods are present out there each with their own pros and cons. The most commonly used techniques for this purpose are statistical, supervised machine learning and graph-based approaches. Statistical methods such as TF-IDF will not go well with this project because in a natural language like English, it would be hard to tell the importance of a word based on factors such as frequency. On the other hand, supervised machine learning techniques were not suitable as the results would tend to reflect the domain in which the model was trained. For example, “computing” would not be a keyword in the mathematics domain, but it would be in the domain of computer science.

Conclusively, graph-based methods were best suited for this project. They work well for keyword extraction because the model isn’t domain specific like in machine learning. The nodes of the graph denote the words while the edges depict the relationship between given word pair. The words that are to be connected vary in the algorithms.

sCAKE (Semantically Aware Keyword Extraction) is one such graph-based, parameter less keyword extraction technique whose experimental results are better than others in the same domain. It creates a Context-Aware Graph (CAG) using co-occurrences of the words. The important thing to note here is that it uses a window size of 2 sentences for measuring co-occurrences. It also proposes a new scoring model for scoring the keywords. Using sCAKE as our base, we propose our own keyword extraction model. This model overcomes the shortcomings of the original algorithm in the following ways: Apart from extracting just keywords, we also focus on keyphrases which serve as a key method of query. We have identified phrases (bigrams and trigrams) using collocations and then added these phrases into the CAG. Along with that, we have used the schwartz and hearst algorithm for replacing abbreviations in the data files for the better functioning of our model. Our model is tagging free as opposed to that of sCAKE. After observation, words of length less than 3 seemed of little importance and hence were discarded. Also, while scoring the words position weight was distributed in the following ratio: 7 for title, 6 for abstract, 1.5 for body and 1 for references. The wordlist formed after keyword extraction is then fed into a word embedding model Glove.

Note that the document in Fig.1. has proper tags for title(--T), abstract(--A) and body(--B):

```
--T
Boolean satisfiability with transitivity constraints.
--A
We consider a variant of the Boolean satisfiability problem where a subset of the propositional variables appearing in formula Fsat encode a symmetric,
transitive, binary relation over N elements. Each of these relational variables, ei,j, for 1  i < j  N, expresses whether or not the relation holds
between elements j.
The task is to either find a satisfying assignment to Fsat that also satisfies all transitivity constraints over the relational variables
(e.g., e1,2 e2,3  e1,3), or to prove that no such assignment exists.
Solving this satisfiability problem is the final and most difficult step in our decision procedure for a logic of equality with uninterpreted functions.
our decision procedure forms the core of our tool for verifying pipelined microprocessors.To use a conventional Boolean satisfiability checker, our augment
the set of clauses expressing Fsat with clauses expressing the transitivity constraints. our consider methods to reduce the number of such clauses based on
the sparse structure of the relational variables.To use Ordered Binary Decision Diagrams (OBDDs), our show that for some sets, the OBDD representation of
the transitivity constraints has exponential size for all possible variable orderings. By considering only those relational variables that occur in the OBDD
representation of Fsat, our experiments show that we can readily construct an OBDD representation
of the relevant transitivity constraints and thus solve the constrained satisfiability problem.
--B
Introduction
Consider the following variant of the Boolean satisfiability problem. our are given a Boolean
formula F sat over a set of variables V . A subset symbolically encodes a binary relation
over N elements that is reflexive, symmetric, and transitive. Each of these relational variables,
whether or not the relation holds between elements
j. Typically, E will be "sparse," containing much fewer than the N(N \Gamma 1)=2 possible variables.
Note that when e i;j 62 E for some value of i and of j, this does not imply that the relation does
not hold between elements i and j. It simply indicates that F sat does not directly depend on the
relation between elements i and j.
```

Fig. 1. The document

An example for the keyphrases extracted from the document is shown below in Fig. 2. where Old_WScore represent the score without position weights, Final Score represents is the score with position weights in account. The words are ranked in decreasing order of score values.

	B	C	D	E	F
1	Words	Old_WScore	Position_Weight	Final Score	
2	boolean-satisfi	21191.27778	5.465226157	153.1946106	
3	formula	40057.43894	1.16208121	61.57407027	
4	transit-constraint-variant	3979.222419	10.5	55.26697804	
5	transit	42393.43658	0.7748625747	43.45117383	
6	relat-variabl	29709.54372	1.08355642	42.5819667	
7	relat	20318.00581	1.302057642	34.99367027	
8	proposit-variabl	15775.82596	1.511709811	31.54559641	
9	express	23883.87445	0.8576982527	27.09676902	
10	problem	37750.96528	0.5349353368	26.71207054	
11	element	18771.13659	1.072062623	26.61882794	
12	subset	8698.815143	2.262155433	26.02919568	
13	epsiv	9941.021018	1.928571429	25.35974749	
14	fsat	12307.75442	1.548076923	25.20284471	
15	encod	17985.61642	0.9226740448	21.95087493	
16	appear	11060.16224	1.290796851	18.88415688	
17	binari	15804.20354	0.8534772907	17.84196934	
18	satisfi-transit-constraint	30378.03912	0.3676600084	14.7735319	
19	decis-procedur	19576.07301	0.4872086498	12.61591547	
20	claus	27330.89381	0.3364079081	12.1618106	

Fig. 2. Top 20 Extracted Key phrases with corresponding Scores

Global Vectors for Word Representation (GloVe) is an unsupervised learning algorithm for obtaining vector representations for words. Model is pre-trained on amassed global word co-occurrences data on a corpus, statistics from a corpus, and the subsequent portrayals feature fascinating results for the word vector space. Cosine similarity between two word-vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbours as indicated by this measurement uncover uncommon however important words that lie outside a normal human's jargon. In this way, in the wake of acquiring the embeddings, Cosine distance is utilized to compute semantic comparability. Following Fig. 3. Shows the embedding for word “Algorithm”.

```
algorithm -0.343093 0.626180 0.171861 0.990223 1.991990 0.822981 0.485791 -0.285670 0.303781 -0.558714 -0
0.230936 0.463966 -0.648849 -0.511597 -0.237017 0.374420 0.462341 2.749464 0.055423 -2.409313 -0.611674 -
0.186697 0.123940 -1.252637 -0.040506 -0.150882 -1.006051 2.275055 -1.575846 -0.692539 -0.821275 0.965382
-0.536988 -0.311801 -0.469580
```

Fig. 3. Embeddings

b. Database Creation

Now we have the keywords defining a document and the embeddings corresponding to each word. We need to map each word to the documents it occurs in and also relate them in some way. For this purpose, we earlier used a knowledge graph, which has the extracted keywords and topics as nodes and the related words have an edge between them. The closer the nodes, the more semantically related they are. We have now replaced the knowledge graph with a database. Earlier, the nodes had some metadata with them indicating the documents the words can be found and where they are located in the filesystem.

Now, two tables are created in the database. First one being the one containing each word, the words closest to it and the inverse of cosine distance between them. This is the same distance that was calculated in the document understanding phase between each word to represent word pair similarity. The second table contains metadata information of each word. Corresponding to each word, the document it belongs to is stored along with its rank in that particular document. The rank here is a representative of the importance of that word in the document. This rank is calculated in the document understanding phase while extracting keywords and keyphrases.

For updating the database whenever a new document is added in the filesystem, once keywords are extracted for the document, entries are made in table 2 at the moment. As for updating the table 1 the whole Glove model will have to be run again to reconstruct word relations. This is a time taking process and hence, instead of updating the table as soon as a new document arrives, it is updated on a regular basis after a certain fixed period of time.

c. Query Understanding

The query entered by the user cannot always be interpreted in the raw form. The user may commit spelling mistakes or the query may contain multiple words or the user may be looking for documents with related words, etc. and hence, Query refining or pre-processing will be required. During the process of Document Understanding, a lot of procedures like stemming, stop word removal are performed on the text which modifies the terms to a great extent. For e.g. ‘Computing’ becomes ‘comput’. To match such words, Query Refinement needs to be performed which include several procedures, which are done in the following order.

First of all, the text beautification is stripped off. The user input is changed to lowercase, punctuation is removed (except hyphen) and numbers are removed. After this, stopword removal is done. The user query is then scrutinised for misspelling using spell checker. Then processed word/sentence after this is separated into individual words for further steps. Then the words are changed into their root words using stemming. We also store a combined version of words(n-grams) along with their individual counterparts (splitting), i.e. 'New York' is saved as 'new', 'york', 'newyork' and 'new-york'. The last step in Query Refinement is finding synonyms of a given word using wordnet. After the pre-processing steps, the modified query is looked into the database for finding related words and corresponding documents.

d. Matching and Ranking

The searched query, once pre-processed would consist of a good number of words. Each word is first matched in table 1 which consists of word pairs and inverse of their cosine difference to locate the searched words neighbours. Once a match is found, the words with closest affinity to the match along with the original word are searched for in table 2. Thus, along with finding exact matches to the word, results will also consist of the closest word those having similar meaning and context to it. It would also display the corresponding documents. The results would further be ranked in terms of relevance and output to the user.

The tables in the database are stored on disk (secondary memory) instead of RAM. For query matching, the words in the table need to be looked up in the procedure described above. Instead of hitting the database tables every time a query needs to be fetched, we have created a dictionary where the query can be matched first. If the matching fails over here then the data from the tables is fetched into the dictionary and then to the user.

The results to the users are ranked in the following order; highest preference is given to trigrams and bigrams keyphrase and its related documents and then comes pre-processed query results along with its metadata information. N-grams are given highest ranking in the results pool generated in response to user query. The top related words fetched from table 1 are displayed in order of closest to farthest. Synonyms are displayed at the end of the ranking ladder to the user. For the word falling within the same pool, they are put in order according to their rank (of keywords) in their respective documents. The ranking shows the most to least relevant documents to the user.

3.2 REQUIREMENT ANALYSIS

The main requirements of the project can be listed in one of the two categories: functional and non-functional.

3.2.1 Functional Requirements

The system shall accept queries from user and return documents containing matching keywords from the query and the documents having semantically similar words. Dataset which we worked on is Krapivin2009 containing full ACM papers. The average document length is 7961. Average number of golden standard key phrases per document is 11 with a standard deviation of 6.44. The data in each document is first refined by removing a pre-defined list of stop words and stemming using Porter -Stemmer. The pre-processing was performed with the help of NLTK, which has various inbuilt functions for these mentioned things. The most essential functional requirements of a search-based project like ours are:

- Data Manipulation – Conversion of dataset to make it easier to read and work upon. In our case, research papers in the form of text files are used and each file consists of separate tags each representing title, body, abstract and references.

- Speed – The time it takes from the moment user enters a query to matching and fetching the results should be as optimal as possible.

- Semantic augmentation – The results apart from containing documents with exact word matching, should also consist of documents containing semantically similar words. This serves as one of the key features of the project.

- Query – The model gives results once a query in the form of a word or phrase is entered by the user. Although there is a spell checker present, the user must avoid the use of noise words in the query.

3.2.2 Non-Functional Requirements

The benchmark used to judge the operations of a project fall under non-functional requirements.

- Usability – The model should be easy to use and user friendly.

- Performance – Calculation time and response time should be as little as possible, because one of any software's features is timesaving.

- Support – If any problem occurs in methods used, it requires code knowledge and natural language processing background to solve.

- Responsiveness and adaptability – The model should be able to adapt to changes and should be growable.

➢ **Hardware Specifications:**

- CPU: Intel i5-6200U CPU @ 2.30GHz * 4
- Graphics: Intel HD Graphics 520(Skylake GT2)
- RAM: 8GB
- Storage: 1TB

➢ **Software Specifications:**

- Python – It is used as the programming language for the project.
- PyCharm – It is one of the IDE used for the implementation of Python program.
- Jupyter Notebook – It is also one of the IDE used for the implementation of Python program.
- Pandas – It is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming.
- Os – This module provides a portable way of using operating system dependent functionality.
- Re – Python's raw string notation for regular expression patterns.
- NumPy – A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- SciPy – It contains modules for optimization, linear algebra, integration, interpolation and special functions.
- Matplotlib – Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- SciKitLearn – It is a Python library widely used for machine learning models. It has inbuilt functions for data models, splitting of data, performing other advanced functions.
- IGraph – Python interface to the igraph high performance graph library, primarily aimed at complex network research and analysis.

- NetworkX – It is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks such as graphs.
- NLTK – A toolkit used for dealing with the text data and it has inbuilt functions to pre-process the data and to convert data in various other formats.
- Pickle – Used for converting an object into stream of characters before saving it to a file.
- MySQL – Used for database management.
- MySQL Connector – To help work with python in MySQL

3.3 SOLUTION APPROACH

For obtaining results for query raised by a user on our local search engine the following steps will be followed: Document Understanding, Database creation, Query Understanding/ Pre-Processing, Query-Document matching and Ranking of matched documents. The Document Understanding incorporates tasks like Keyword/Keyphrase extraction and vector representation of words. Database creation involves making of two tables, one for word to word affinity and other for metadata.

Automatic identification of words or phrases which best describe a given document is known as keyword/keyphrase extraction technique. There are different types of keyword extraction techniques present today. We focused on techniques that were graph based. Machine learning techniques like Supervised learning required the manual labour of annotating the data which make our model dependent on expert knowledge.

Keywords are required to act as terms for the database creation. A context aware graph (CAG) is made using 2 consecutive sentences as the window size. First and foremost, abbreviations are identified and replaced in the data following which text pre-processing is done by stop word removal and stemming the documents. NLTK is used for most of the refining task. Each window is treated as a document and term document frequency matrix is obtained. By multiplying this matrix with its transpose, we get the term-term frequency with the help of which the graph is created. Apart from the individual terms, bigrams and trigrams are also added based on collocations. Words are ranked in the final CAG using four different metrics, namely:

- Semantic Strength: Robustness of connection between two words W_i and W_j is marked by the number of times the two words co-occur in same context, and is elicited by weight v_{ij} of edge e_{ij}.
- Position Weight: Words occurring first in the document are given more weight than words that occur later. Weights are assigned to words occurring in title, abstract, body and references accordingly.

- Level of Hierarchy: Instead of clustering, trussness is used to mark hierarchy of words in the graph.
- Semantic Connectivity: The scope of semantic connectivity of a word is measured by examining the number of distinct concepts that it is linked with.

The next step towards the construction of database is to find term-term similarity which represents the edges of the graph. Word embedding is a feature learning technique in natural language processing (NLP) which takes the words and returns a vector of real numbers representing the relation between terms. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension. There are various ways of performing word embedding which include neural networks, dimensionality reduction on the word co-occurrence matrix, probabilistic models etc. We have studied models like Glove [16], BERT [18], word2vec [15] etc. Glove is used along with Cosine distance to compute this similarity.

Glove is run globally over all the database documents to create embeddings. The database is created using inputs from keyword extraction and word embeddings MySQL is used for database creation in which all keywords are entries of table 1 along with their closest words and along with the metadata and ranks in table 2. Table 2 would help in mapping the searched word back to its corresponding document. Resultant terms from Query Understanding will be matched in the database. The result of the matched Query-document pair will be then ranked according to their priority and will be displayed to the user.

4. MODELLING AND IMPLEMENTATION DETAILS

4.1 DESIGN DIAGRAMS

4.1.1 Use Case Diagram

The following diagram shows the procedure of interaction of user with model. Whenever a user enters a query in the form of a word or a phrase, the query is pre-processed and the resultant words are searched for in the database. When the matching word is found, its connections are retrieved and the corresponding metadata is also fetched. The final output is shown to the user in the form of the words in the decreasing order of relevancy along with their information, i.e. file to which they belong and the rank.

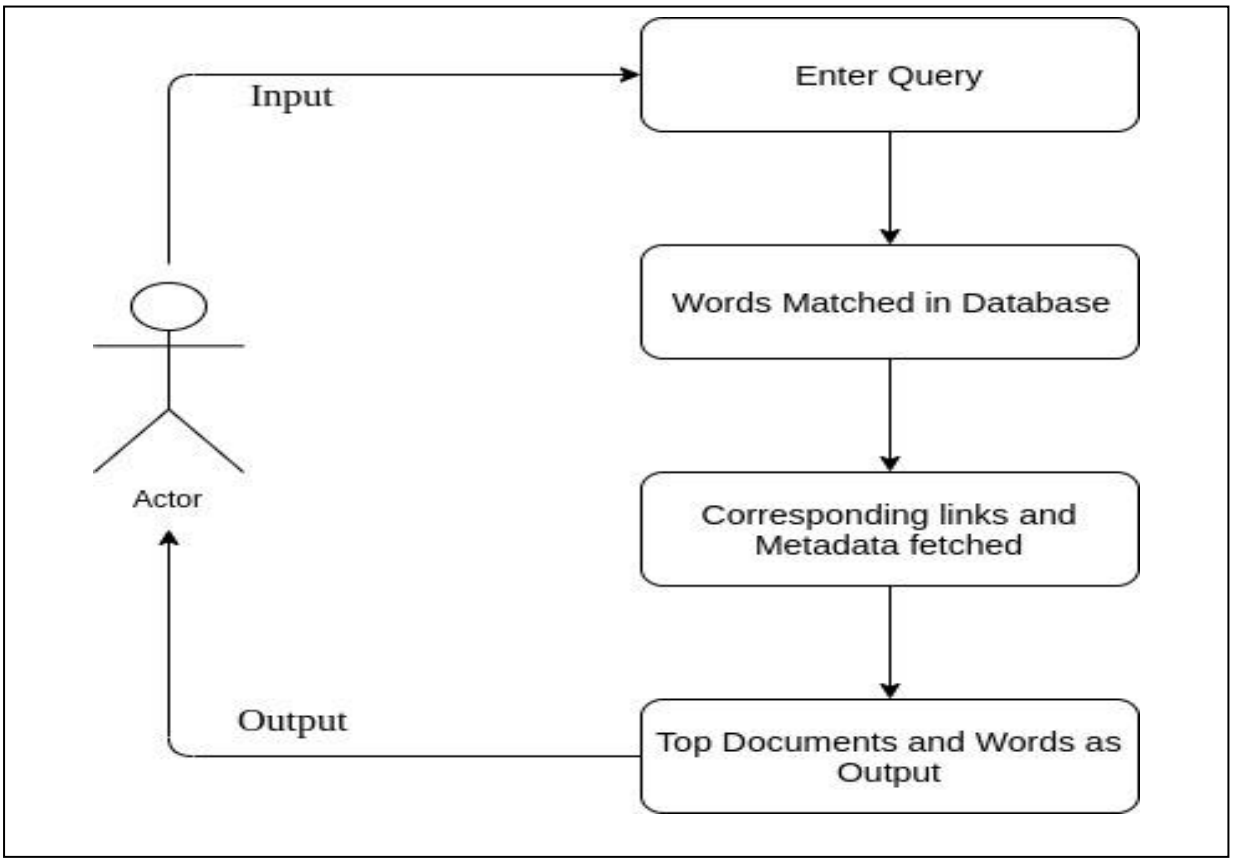


Fig. 4. Use Case Diagram

4.1.2 Control Flow Diagram

The following diagram show the flow of the entire model. The first step is Document Understanding followed by Database Creation, Query Understanding and retrieval of relevant information, ranking of documents and Displaying results.

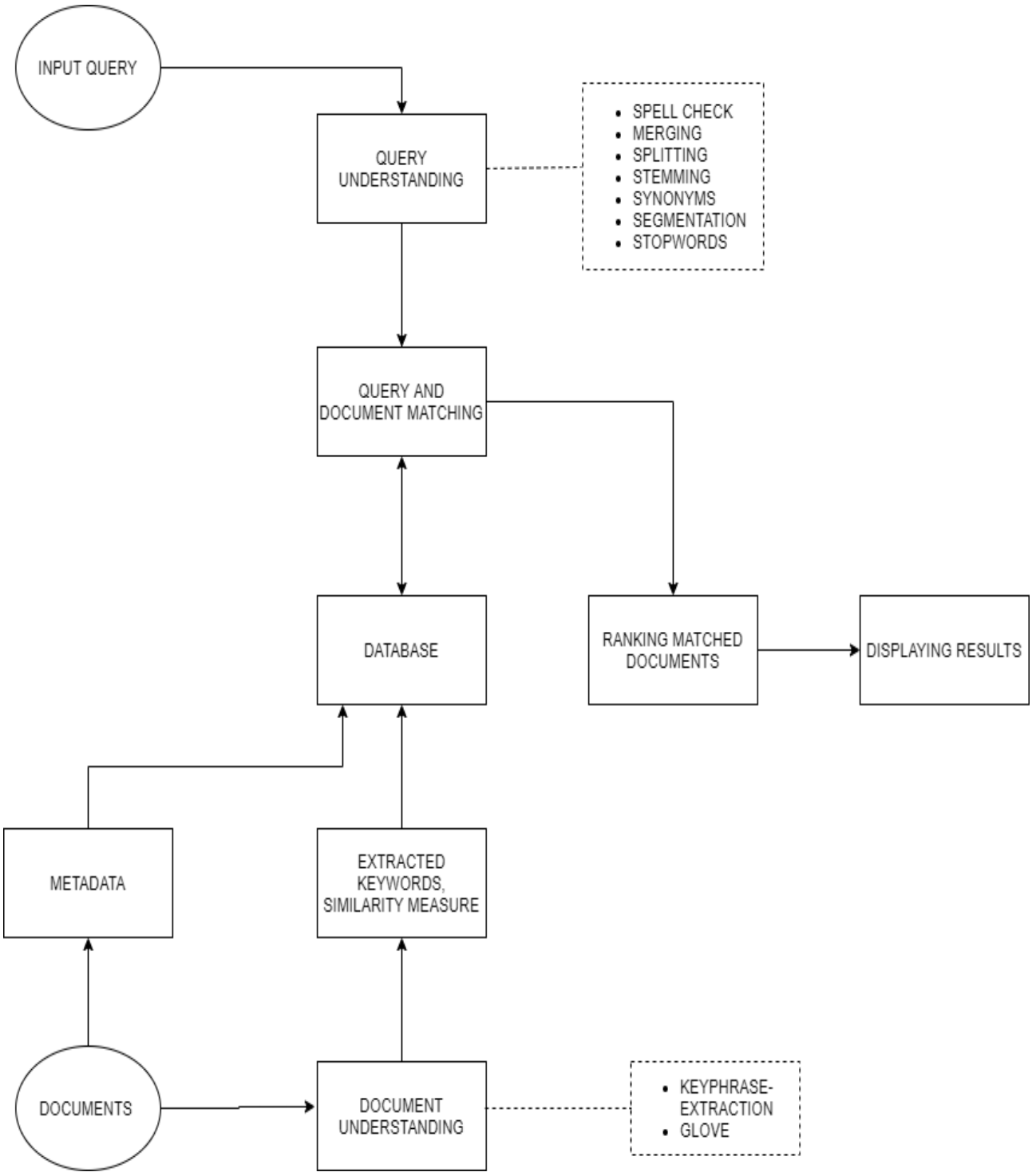


Fig. 5. Control flow diagram

The following flow diagrams detail the keyphrase extraction process and the matching process. For keyphrase extraction once the dataset is pre-processed and given appropriate position weights, the Context Aware Graph (CAG) is created using a window size of 2 sentences. Evaluating properties of graph like trussness, each word is given a score and ranked results are then stored. The matching process proceeds as follows: Once a query is refined, the corresponding words are searched in the dictionary. This dictionary has recently searched words as keys and pointer values of where the word is stored in a doubly linked list as corresponding values. If found, the respective node is moved to the head of link list and the results are retrieved and displayed. If not found, the words are searched in database and entries are made into the dictionary and linked list accordingly.

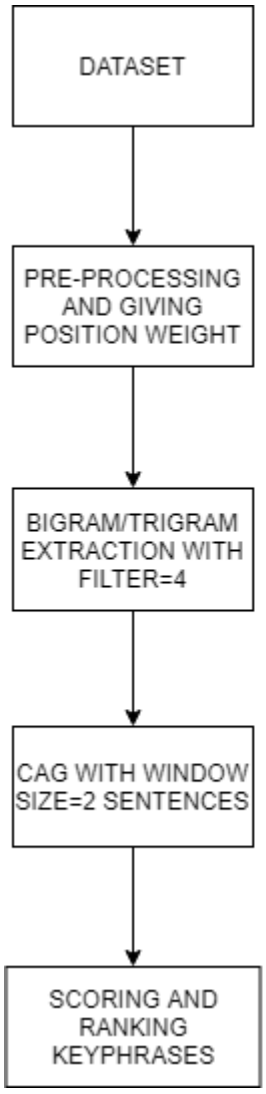


Fig. 6. Extraction of Keyphrases

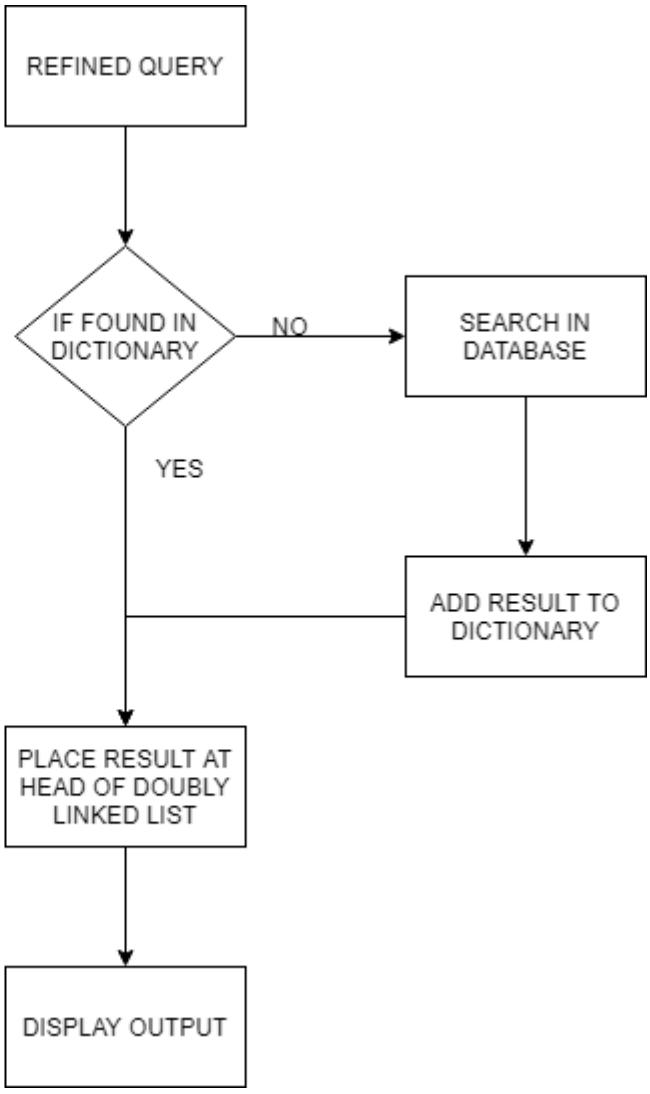


Fig. 7. Matching and fetching results

4.1.3 Sequence Diagram

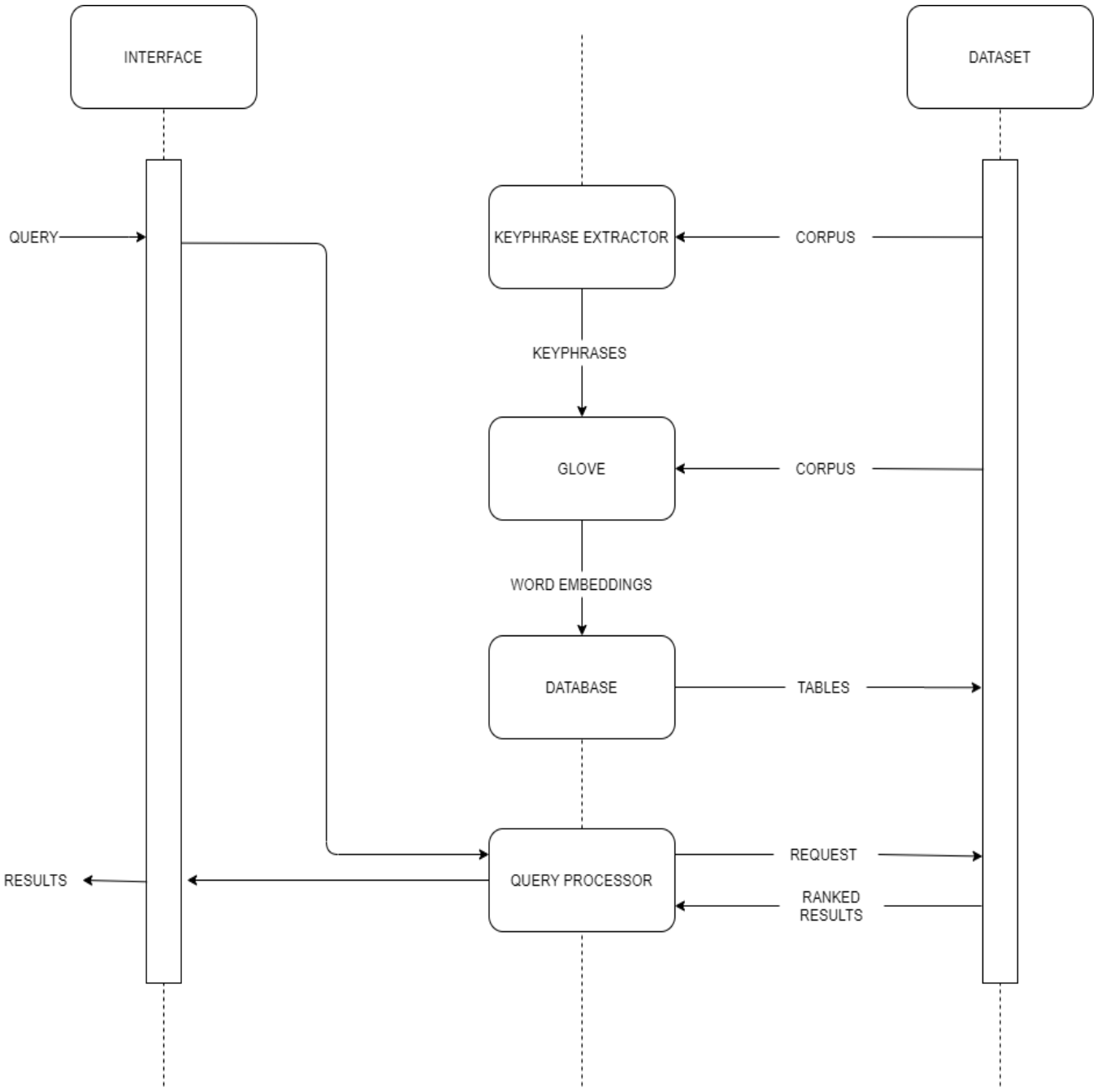


Fig. 8. Sequence Diagram

4.2 IMPLEMENTATION DETAILS

- **Dataset:**

  We worked on the benchmark dataset which we worked on is Krapivin2009 containing full ACM papers. The average document length is 7961. Average number of golden standard key phrases per document is 11 with a standard deviation of 6.44. The data in each document is first refined by removing a pre-defined list of stop words and stemming using Porter -Stemmer. The pre-processing was performed with the help of NLTK, which has various inbuilt functions for these mentioned things.

- **Keyphrase Extraction:**

  The documents are fed into the code in text form with proper tags to identify title, abstract, body and references and the following order is executed to form resultant graph and extracted keywords:

  - Apply schwartz hearst algorithm for abbreviations
  - Create Position Information
  - Create the Context Aware Graph
  - Influence Evaluation of a node in the graph
  - Word Score with Position Weight: Final score of each word is calculated by multiplying resultant score from influence evaluation and position weight. These results are stores in a csv file.

  The results are stored into separate documents for graph, score and keywords/keyphrases.

- **Word Embedding:**

  To represent words in the vector, glove (Global Vectors for Vector Representation) by Stanford is being used. The standard glove module with its pre-trained model is used to convert the extracted key phrases from each document into a vector from. The resultant vector is of 50 dimensions. For finding out the cosine similarity between every word, a python code is implemented.

- **Database and Metadata:**

  Two tables are created, namely wordSim and wordDoc. Table 1 is for storing word pairs and inverse of their cosine distance and Table 2 is for storing metadata. The schemas for tables created in database using MySQL are shown in Fig. 9.

```
mysql> describe wordSim;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Word1    | varchar(100) | YES  |     | NULL    |       |
| Word2    | varchar(100) | YES  |     | NULL    |       |
| Distance | float        | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.04 sec)

mysql> describe wordDoc;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| Word  | varchar(100) | YES  |     | NULL    |       |
| Doc   | varchar(200) | YES  |     | NULL    |       |
| Rank  | int(11)      | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Fig. 9. Schema for database tables

- **Query:**

  The query is entered by user and relevant words are displayed accordingly. Many processes of query refinement are included in the model. NLTK and Autocorrect libraries are used for doing most of the refining work.

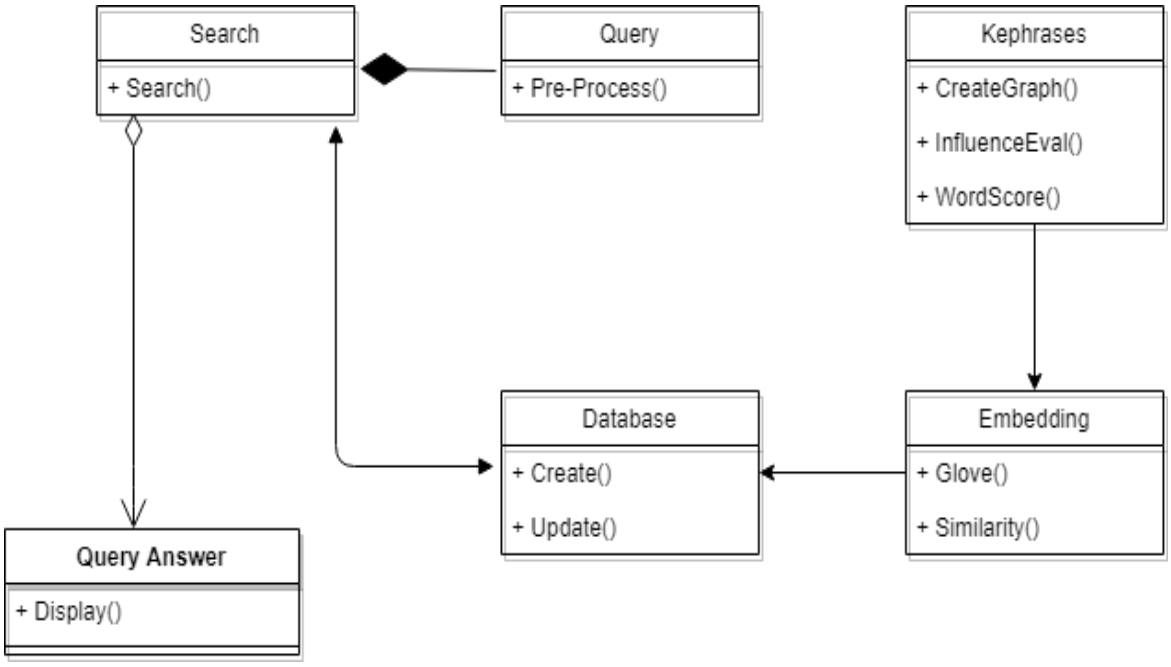


Fig. 10. Implementation details of modules

- **Issues encountered:**
  - Finding and implementing a suited technique for keyphrase extraction. Extensive search to model the final algorithm.
  - Maintaining the versions of the dependencies used in the project. We eventually created a virtual environment and installed the dependencies within.
  - While coding the collocations, the data resulted in an encoded form, we had to decode it to form the bigrams and trigrams.
  - Since there is an often use of abbreviations in a document representing some important word, the important words were getting less score. We used schwartz and hearst algorithm to resolve this.
  - Earlier we were using a Knowledge graph to store semantic connections. It had several issues as it was saved in the memory. It was prone to corruption; inconsistent data if the device shuts down and was insufficient in handling failures. We are now using a database which stores data on the disc.

4.3 RISK ANALYSIS

Some of the common risks associated with our project are listed as follows:

- Data Shortage: If the user doesn't enter a word present in the vocabulary then chances are there that you don't get the desired results. That is the reason why our dataset contains this many numbers of research papers.

- Accuracy: Since Machine Learning is probabilistic approach, there are high chances that the accuracy may vary. The mitigation for this risk is again the size of the data as presented by many researchers.

- Epochs Used in the model: There are chances that changing the epochs in the GloVe model may change the accuracy, hence one has to pick each epoch very judiciously.

Following risks are posed by various agents on the model. For impact we have used High, Medium and Low as the ratings giving a score of 5, 3 and 1 for each. The risk factors are illustrated in Table 1. And the mitigation plan corresponding to those factors can be found in Table 2.

Table 1. Various Risk Factors

Risk Id	Classification	Description	Risk Area	Impact	Probability	RE(P*I)
1.	Performance	A performance risk is the potential not to provide as much value as a product, service, system or project needs.	This can apply to internal modules or the result of a query search if the expected results are not given.	5	0.5	2.5
2.	Hardware and Software	Hardware and Software issues are common causes of data problems.	Power may fail, memory may not be adequate, software may be installed incorrectly or may not be installed at all, you can mistype, there are accidents of all kinds.	3	0.2	0.6
3.	Maintainability	Whenever a new document is added to the dataset or a present document is modified.	The code requires a rerun to make semantic connections.	1	0.4	0.4
4.	Scale	Failure to manage capacity and scalability risk.	Service failures or performance degradations due to lack of capacity; under increased workloads, the architecture may break down.	3	0.3	0.9
5.	Difficulty	If it is an amateur user using the code, he/she might face problems getting used to it and applying it to daily use.	Difficulty might occur in dealing with the code or indexing the documents.	1	0.5	0.5

Table 2. Mitigation Plan

Risk Id	Mitigation Plan
1.	Before putting the project in place, the background was studied extensively and the framework to be worked on was well understood.
2.	To prevent memory or shutdown failures, we have used a database so that all data is stored on disk instead of memory.
3.	There are two tables present in the database. One of them will be updated as soon as a new document is added, the other one updates regularly after a certain period of time.
4.	The project will be based on systems with strong processing power and features needed to make our model operate efficiently.
5.	A simple UI can be provided for the user to operate the model more easily along with a documentation on how to work through it.

5. TESTING

5.1 TESTING PLAN

There are many testing types used in the testing phase. Some of them are listed below:

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Stress Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing
- White Box Testing
- Load Testing

Out of all the testing types, we used Unit testing, Integration testing, Requirements testing and Load testing. First and foremost, requirement testing was done. It is important to lay down a requirement analysis step before starting coding to avoid any bugs in the further phases of the project. Whenever a module was completed, we used Unit testing at that module of the project. Once the code started coming together, we went for Integration testing on the model construction part. Load testing and performance testing were performed simulated. A table (Table 3) enlisting all the tests and software components along with brief description is shown below.

Table 3. Testing Plan

Type of Test	Is test Performed?	Explanations	Software Component
Requirement Testing	Yes	This was for checking whether all the functional and non-functional requirements such as all modules were working or not.	Testing performed on basis of heuristics like data, platform and operations.

Unit Testing	Yes	The individual units of the code such as Graph Creation, Scoring and Word embedding were up and running.	The unit testing was performed on the functions of the code.
Integration Testing	Yes	All the modules were working properly when run together.	While integrating machine learning model like Glove with other algorithm-based techniques, all modules were working in perfect sync. (Manual Testing)
Performance Testing	Yes	To assess the speed, responsiveness and stability of the project.	No module was found that holds back overall performance.
Load Testing	Yes	To determine behaviour of the model when run under certain load.	The code was run on a dataset of 2304 documents to check if it could handle huge amount of data.

5.2 COMPONENT DECOMPOSITION

Testing can be performed on keyphrase extraction phase, word embedding, database, query phase and the project as a whole. Which component needs testing and the type of testing required are enlisted below:

- **Module:** Keyword/Keyphrase Extraction

  Types of Testing: Integration Testing, Load Testing, Performance Testing

  Description: The keyword extraction consists of various sub-modules and hence, all of them need to run when connected together. The module as a whole should perform well on a huge amount of data and as this module forms the base of the project, the keywords extraction must be accurate. They must match the theme of the document.

- **Module:** Word embedding

  Types of Testing: Unit Testing

  Description: It works as a separate module of the project converting keywords into vector representations and hence requires unit testing.

- **Module:** Database

  Types of Testing: Database Testing, Load testing

  Description: This is to ensure that the schema, tables etc. are being built correctly. The updated values should be reflected in the database. The database should be able to handle large entries appropriately.

- **Module:** Query

  Types of Testing: Manual Testing

  Description: The user entered query should be accurately pre-processed after which matching with database will take place. Since there was no database available to test the results, a manual testing had to be done.

5.3 TEST CASES

The following table shows the test cases mentioned in the component decomposition section and their status.

Table 4. Test Cases

Test Case Id	Input	Expected Outcome	Status
1.	Document related to machine learning for Key phrase extraction.	Machine Learning, Supervised/Unsupervised, Neural Network, Artificial Intelligence. Good recall and f1 values.	PASS

2.	Document input for word embedding.	A 50-dimension vector representation of each word.	PASS
3.	Metadata Storage	Database table updated whenever a new document is encountered.	PASS
4.	Word or phrase in the form of query	Exact word match and related words should show up	PASS
5.	Noise in the query	Exact match and related words should show up	FAIL

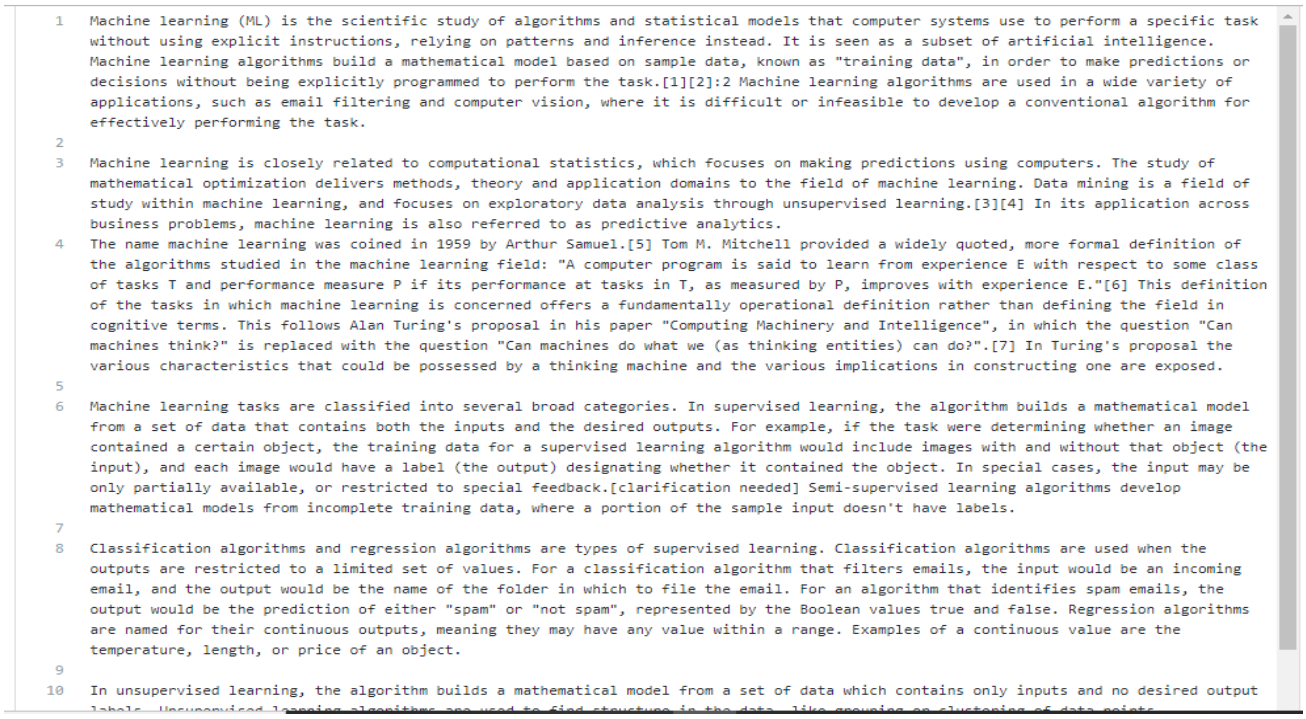


Fig. 11. Test case for keyphrase extraction- A ML based paper

Search this file...

		Old_WScore	Position_Weight	SCScore	Words
2	2	6114.876543209875	0.47208575454291435	2886.7461068380017	algorithm
3	165	1977.3508771929826	1.0	1977.3508771929826	scientif
4	25	3642.7669753086416	0.5387396484739356	1962.5029997502393	studi
5	3	5639.1296296296305	0.26126280161935034	1473.2948057317267	model
6	6	5493.024691358024	0.24276918663679195	1333.5371364968025	comput
7	1	6153.3703703703695	0.15260458890416836	939.0325557454605	machin-learn
8	31	3455.112318840579	0.2672413793103448	923.3489817591202	statist
9	11	5104.939814814815	0.16588411102287184	846.8284030058196	task
10	19	3862.6010802469127	0.19034740749823476	735.2361018248808	perform
11	7	5401.938697318008	0.13149634085981043	710.3351722463291	data
12	0	6568.98688271605	0.06888199325397101	452.48491014067105	learn
13	9	5139.296934865901	0.07601679614852014	390.67288744441555	mathemat
14	21	3771.7763157894733	0.07496214033316506	282.74042548951877	pattern
15	164	1977.3508771929826	0.14285714285714285	282.47869674185466	system
16	16	4396.781609195402	0.06407156261366967	281.70866817219445	predict
17	4	5599.757575757576	0.04303699318114863	240.9967286039642	input
18	15	4412.607279693487	0.05416247453450646	238.99772941717632	train
19	160	1977.3508771929826	0.1111111111111111	219.7056530214425	specif

Fig. 12. Corresponding output with all the Scores- Related words showing up

5.4 ERROR AND EXCEPTION HANDLING

- CASE I: Passing a smaller number of features than required

  If the document has words only of length less than three, or it is an empty document. The output will consist of only bigrams and trigrams in first case and nothing in the second case.

- CASE II: Passing data in inappropriate format

  The original research papers are in pdf format but the code accepts documents in csv format. That needs to be taken care of.

- CASE III: Passing empty string

  If the user enters an empty string, then nothing will be displayed.

- CASE IV: Passing out of vocabulary string

  If the query entered by user does not match any word after pre-processing maybe because of wrong spelling, it is termed as an out of vocabulary word.

5.5 LIMITATIONS OF THE SOLUTION

This model does not read pdf(s) and requires their conversion into text files beforehand. Also, the text file should contain appropriate headers for title, abstract, body and references else the model won't function correctly. Changes made in the text document after the database is built can't be read and therefore aren't reflected in this model. Once a new document is added to the dataset, table 1 in the database is not updated simultaneously. Since it is a rather lengthy and time taking task, a certain amount of period is it set after which the database is updated and new changes are reflected in it. The model also fails to capture the deletion of files action as of this moment. Also, for finding out word similarity, apart from Cosine distance other measures can be used or multiple methods can be integrated together. One limitation is when user enters some gibberish, the spell checker embedded in the query refinement code might form it into a logical word and show results which aren't related. We need to find an alternative for the same.

6. FINDINGS, CONCLUSION AND FUTURE WORKS

6.1 FINDINGS

We run our code for the Krapivin dataset which contains full ACM papers. The comparison is done with sCAKE algorithm using metrics like recall, precision and f1-score for the keywords/keyphrase extraction process.

After introducing the changes mentioned above in the keyword/keyphrase extraction algorithm, we did a comparison of sCAKE algorithm and our extension to it. This was done using f1-score values. We used 'k' words from the extracted keywords to match with golden standard data, ranging 'k' from 1 to 50. Upon executing sCAKE and our algorithm, following results were observed:

- We have a better max f1-score value: 45.587 which results at a value of k=4 as compared to that of sCAKE that is 40.33 at a value of k=5.
- Our results are better than sCAKE for the following ranges of k: 1-50

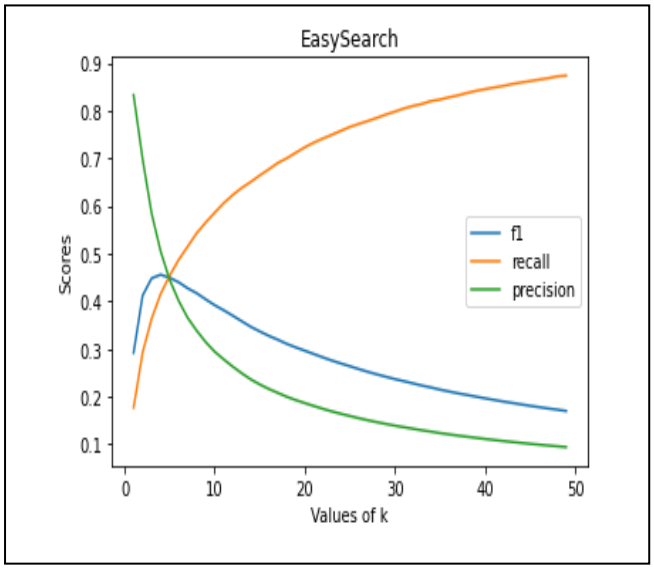
k	recall	precision	f1	max_f1	max_f1_at
1	0.176951	0.832899	0.291889	0.291889	1
2	0.292063	0.698785	0.411949	0.411949	2
3	0.363532	0.584635	0.448304	0.448304	3
4	0.415413	0.5051	0.455887	0.455887	4
5	0.453809	0.445313	0.449521	0.455887	4
6	0.487172	0.401114	0.439975	0.455887	4
7	0.514849	0.365327	0.427388	0.455887	4
8	0.543134	0.338433	0.417018	0.455887	4
9	0.565366	0.314477	0.404151	0.455887	4
10	0.58608	0.294314	0.391851	0.455887	4
11	0.605991	0.278093	0.381235	0.455887	4
12	0.623242	0.262948	0.369854	0.455887	4
13	0.638011	0.249032	0.358235	0.455887	4
14	0.650569	0.236173	0.346542	0.455887	4
15	0.664169	0.22555	0.336743	0.455887	4
16	0.677073	0.216064	0.32759	0.455887	4
17	0.690428	0.207797	0.31945	0.455887	4
18	0.700231	0.199629	0.310684	0.455887	4
19	0.71195	0.19264	0.303231	0.455887	4
20	0.723267	0.18635	0.296346	0.455887	4
21	0.733055	0.180266	0.289373	0.455887	4
22	0.74167	0.174361	0.282345	0.455887	4
23	0.749596	0.168875	0.275649	0.455887	4
24	0.757872	0.163882	0.269489	0.455887	4
25	0.76601	0.159306	0.263758	0.455887	4
26	0.772415	0.154664	0.257723	0.455887	4

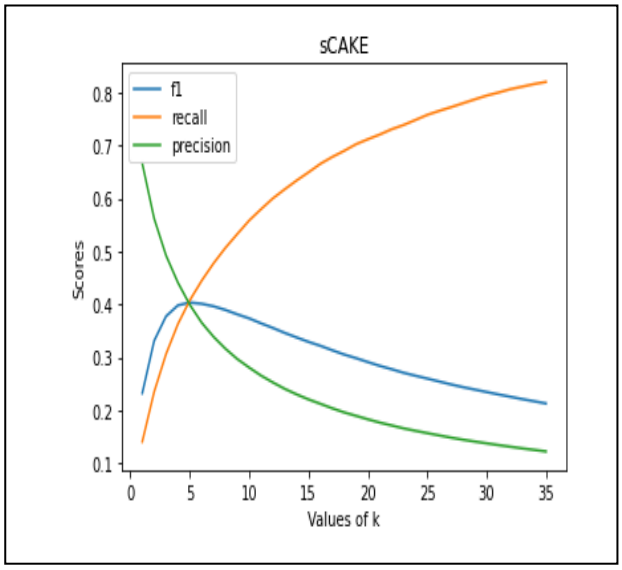
Fig. 13. EasySearch keyphrase extraction results

k	recall	precision	f1	max_f1	max_f1_at
1	0.140597	0.665799	0.232167	0.232167	1
2	0.234667	0.561849	0.331061	0.331061	2
3	0.305965	0.492188	0.377352	0.377352	3
4	0.362625	0.440972	0.397979	0.397979	4
5	0.40747	0.399219	0.403302	0.403302	5
6	0.444423	0.365234	0.400956	0.403302	5
7	0.477274	0.338666	0.396197	0.403302	5
8	0.506522	0.316135	0.389298	0.403302	5
9	0.532624	0.296634	0.38105	0.403302	5
10	0.558163	0.280512	0.373378	0.403302	5
11	0.57953	0.265349	0.364023	0.403302	5
12	0.6	0.252387	0.355313	0.403302	5
13	0.617173	0.240284	0.3459	0.403302	5
14	0.63377	0.229849	0.337351	0.403302	5
15	0.649178	0.22037	0.329043	0.403302	5
16	0.665183	0.21205	0.321584	0.403302	5
17	0.678215	0.203585	0.313164	0.403302	5
18	0.689653	0.195891	0.305116	0.403302	5
19	0.702216	0.189373	0.298301	0.403302	5
20	0.711631	0.182791	0.290869	0.403302	5
21	0.72051	0.176629	0.283708	0.403302	5
22	0.730477	0.171224	0.27742	0.403302	5
23	0.738464	0.165723	0.270697	0.403302	5
24	0.747918	0.161097	0.265094	0.403302	5
25	0.757614	0.156719	0.259714	0.403302	5
26	0.76515	0.152427	0.254212	0.403302	5

Fig. 14. sCAKE keyword extraction results

Metrics variation with value of ‘k’





Graph 1. EasySearch Keyphrase Extraction

Graph 2. sCAKE Keyphrase Extraction

- **Query Results:**

```
Search: positive semidefinite hessian

posit-semidefinit
/home/nayan/coding/Major/EasySearch/venv/data/589159.txt
/home/nayan/coding/Major/EasySearch/venv/data/588960.txt

posit
/home/nayan/coding/Major/EasySearch/venv/data/604133.txt
/home/nayan/coding/Major/EasySearch/venv/data/587409.txt
/home/nayan/coding/Major/EasySearch/venv/data/506843.txt
/home/nayan/coding/Major/EasySearch/venv/data/507059.txt
/home/nayan/coding/Major/EasySearch/venv/data/196757.txt

semidefinit
/home/nayan/coding/Major/EasySearch/venv/data/596141.txt
/home/nayan/coding/Major/EasySearch/venv/data/587812.txt
/home/nayan/coding/Major/EasySearch/venv/data/587192.txt
/home/nayan/coding/Major/EasySearch/venv/data/603891.txt
/home/nayan/coding/Major/EasySearch/venv/data/350710.txt

hessian
/home/nayan/coding/Major/EasySearch/venv/data/603891.txt
/home/nayan/coding/Major/EasySearch/venv/data/350710.txt
/home/nayan/coding/Major/EasySearch/venv/data/598459.txt
/home/nayan/coding/Major/EasySearch/venv/data/301669.txt
/home/nayan/coding/Major/EasySearch/venv/data/587201.txt

posit-definit
/home/nayan/coding/Major/EasySearch/venv/data/587883.txt
/home/nayan/coding/Major/EasySearch/venv/data/587865.txt
/home/nayan/coding/Major/EasySearch/venv/data/587192.txt
/home/nayan/coding/Major/EasySearch/venv/data/350710.txt
/home/nayan/coding/Major/EasySearch/venv/data/301669.txt

symmetr-matrix-posit
/home/nayan/coding/Major/EasySearch/venv/data/587895.txt

complementar-nondegeneraci
/home/nayan/coding/Major/EasySearch/venv/data/589287.txt

complet-problem-posit
/home/nayan/coding/Major/EasySearch/venv/data/587872.txt

jacobian
/home/nayan/coding/Major/EasySearch/venv/data/272885.txt
/home/nayan/coding/Major/EasySearch/venv/data/289843.txt
/home/nayan/coding/Major/EasySearch/venv/data/359222.txt
/home/nayan/coding/Major/EasySearch/venv/data/349174.txt
/home/nayan/coding/Major/EasySearch/venv/data/350710.txt

state
/home/nayan/coding/Major/EasySearch/venv/data/192888.txt
/home/nayan/coding/Major/EasySearch/venv/data/318943.txt
/home/nayan/coding/Major/EasySearch/venv/data/590522.txt
/home/nayan/coding/Major/EasySearch/venv/data/593360.txt
/home/nayan/coding/Major/EasySearch/venv/data/587409.txt
```

Fig. 15. Results for – "Positive semidefinite hessian"

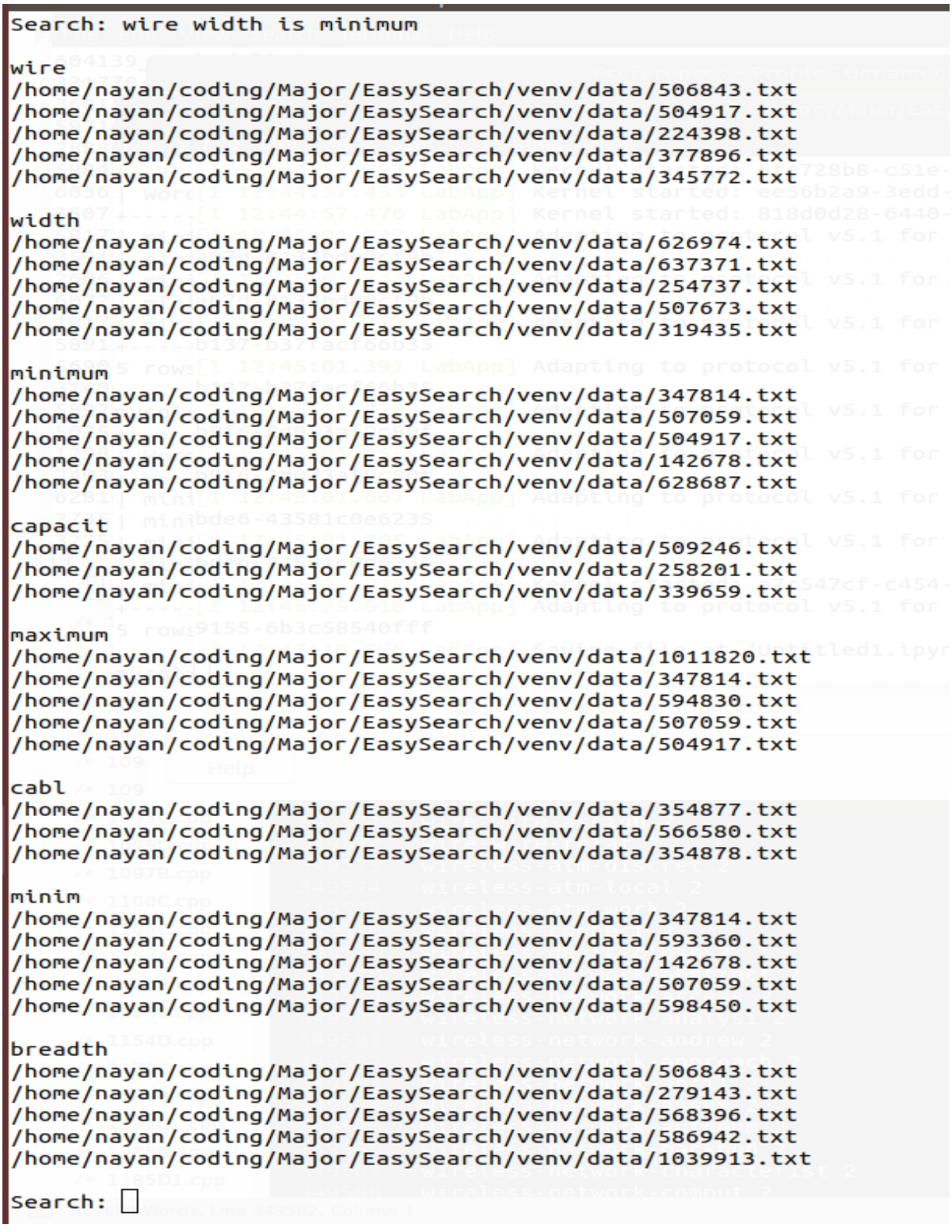


Fig. 16. Results for-“wire width is minimum”

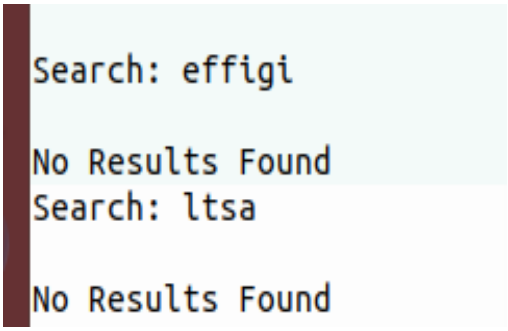


Fig. 17. Results for Noise words

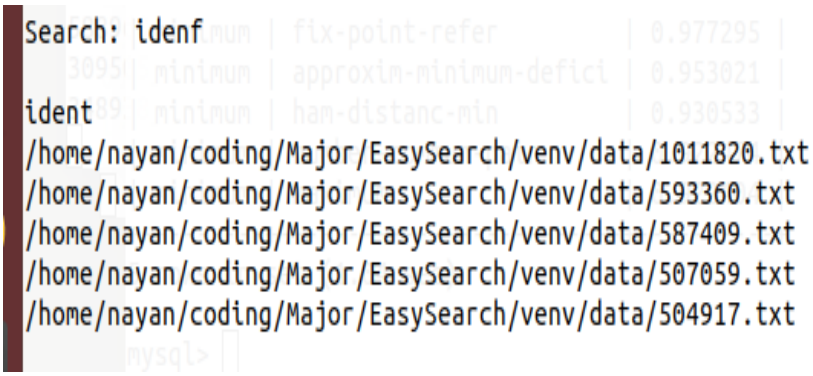


Fig. 18. Word is Spell checked

6.2 CONCLUSION

After extensive survey we came across different techniques for the planned sub-modules for our semantic search engine. The techniques we studied were diverse in nature and offered us with alternate viewpoints to deal with the given problem. For keywords/keyphrase extraction we settled upon graph-based approach of sCAKE as it provided us with best results than its predecessors. We implemented the same by introducing various modifications like addition of bigrams and trigrams, abbreviation resolution, weighted positions, a tagging free approach etc. Once the final model for keyword/keyphrase extraction was completed and it gave satisfactory results, the process of database creation began. As discussed, GloVe was used as the word embedding technique along with a suitable word similarity metric, cosine distance. The database created consists of two tables, one for storing similar words and another for storing metadata. Alongside, several procedures were adapted to refine the query entered by user. This was done to expand the scope of matching more similar words. Once the document understanding, query refining and database creation procedures were over, matching and similar words retrieval was done. Speedy access of results was enabled with the help of a doubly linked list and a dictionary. Finally, the results are ranked and output is displayed.

6.3 FUTURE WORKS

The work done till now consists of keyphrase extraction, vector representation of words and making of database along with metadata storage and matching and ranking a query. Future work will focus on handling the shortcomings of the project. As of now, the model does not read pdf(s) and demands that they be translated to text files in advance. The text file should also contain correct identifiers for title, abstract, body and references otherwise the model does not work as intended. These can be overcome by implementing a pdf reader and identify tags while reading the document. Also, the model fails to remove metadata and words from the table related to a deleted file. This can be covered in future. Further the performance of the keyphrase extraction can be improved by introducing new changes. In addition, other measures may be used to find word similarity apart from Cosine distance, or multiple methods may be assimilated together. Integration of the whole model with the Operation System and development of a suited User Interface are also some of the developments that can be made into the project.

REFERENCES

1. Siddiqi, S., & Sharan, A. (2015). Keyword and keyphrase extraction techniques: a literature review. *International Journal of Computer Applications*, *109*(2).
2. Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into text. In EMNLP'04, 404–411.
3. Wan, X., & Xiao, J. (2008, July). Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *AAAI* (Vol. 8, pp. 855-860).
4. Rose, Stuart & Engel, Dave & Cramer, Nick & Cowley, Wendy. (2010). Automatic Keyword Extraction from Individual Documents. 10.1002/9780470689646.ch1.
5. Dutta, Ambar. (2016). A Novel Extension for Automatic Keyword Extraction. International Journal of Advanced Research in Computer Science and Software Engineering. 6. 160-163.
6. Litvak, M., Last, M., Aizenman, H., Gobits, I., & Kandel, A. (2011). DegExt—A language-independent graph-based keyphrase extractor. In *Advances in Intelligent Web Mastering–3* (pp. 121-130). Springer, Berlin, Heidelberg.
7. Rousseau, F., Vazirgiannis, M., 2015. Main Core Retention on Graph-of-Words for Single-Document Keyword Extraction, in: Advances in Information Retrieval. Springer, pp. 382–393.
8. Florescu, C., Caragea, C., 2017. A Position-Biased PageRank Algorithm for Keyphrase Extraction, in: AAAI, pp. 4923–4924.
9. Duari, S., & Bhatnagar, V. (2019). sCAKE: Semantic Connectivity Aware Keyword Extraction. *Information Sciences*, *477*, 100-117.
10. Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*(2-3), 259-284.
11. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.
12. Zhao, W., Chen, J. J., Perkins, R., Liu, Z., Ge, W., Ding, Y., & Zou, W. (2015, December). A heuristic approach to determine an appropriate number of topics in topic modeling. In *BMC bioinformatics* (Vol. 16, No. 13, p. S8). BioMed Central.
13. Nikolenko, S. I., Koltcov, S., & Koltsova, O. (2017). Topic modelling for qualitative studies. *Journal of Information Science*, *43*(1), 88-102.
14. Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" : *Mining of Massive Datasets*. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2.

15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
16. Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).
17. Christopher Moody, 2016, Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec
18. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
19. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, 2018. Deep contextualized word representations.
20. Chirita, P. A., Costache, S., Nejdl, W., & Paiu, R. (2006, June). Beagle++: Semantically enhanced searching and ranking on the desktop. In *European Semantic Web Conference* (pp. 348-362). Springer, Berlin, Heidelberg.
21. ZHOU, Q., LE, X. Q., & LI, X. (2013). Applications of JNI Technology in Desktop Search Tool. *Computer Technology and Development*, *2*.
22. Kenter, T., & De Rijke, M. (2015, October). Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 1411-1420). ACM.
23. Richardson, R., Smeaton, A., & Murphy, J. (1994). Using WordNet as a knowledge base for measuring semantic similarity between words.
24. Opsahl, T., & Panzarasa, P. (2009). Clustering in weighted networks. *Social networks*, *31*(2), 155-163
25. Balakrishnan, V., & Lloyd-Yemoh, E. (2014). Stemming and lemmatization: a comparison of retrieval performances.
26. Kiela, D., Hill, F., & Clark, S. (2015, September). Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 2044-2048).
27. McKeown, K. R., & Radev, D. R. (2000). Collocations. *Handbook of Natural Language Processing. Marcel Dekker*, 1-23.
28. Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, *68*(13), 13-18.
29. Machine learning for entity coreference resolution: A retrospective look at two decades of research." *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

30. Wiseman, Sam, Alexander M. Rush, and Stuart M. Shieber. "Learning global features for coreference resolution." *arXiv preprint arXiv:1604.03035* (2016).
31. Li, Weijia, et al. "Dynamic hashing: Adaptive metadata management for petabyte-scale file systems." *23rd IEEE/14th NASA Goddard Conference on Mass Storage System and Technologies*. 2006.
32. Martín, Germán Hurtado, et al. "Metadata impact on research paper similarity." *International Conference on Theory and Practice of Digital Libraries*. Springer, Berlin, Heidelberg, 2010.
33. Wu, Yi-fang Brook, et al. "Domain-specific keyphrase extraction." *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005.
34. Hagberg, Aric, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. No. LA-UR-08-05495; LA-UR-08-5495. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
35. Paulheim, Heiko. "Knowledge graph refinement: A survey of approaches and evaluation methods." *Semantic web* 8.3 (2017): 489-508.
36. Beliga, S. (2014). Keyword extraction: a review of methods and approaches. University of Rijeka, Department of Informatics, 1-9.
37. Schwartz, A. S., & Hearst, M. A. (2002). A simple algorithm for identifying abbreviation definitions in biomedical text. In *Biocomputing 2003* (pp. 451-462).
38. Gawlik, M., & Strable, C. (2010). Comparison of abbreviation recognition algorithms.
39. Li, H., & Xu, J. (2014). Semantic matching in search. *Foundations and Trends® in Information Retrieval*, *7*(5), 343-469.

URVI AGARWAL

Jaypee Institute of Information Technology, Sector 62, Noida, India
8630843598 | agarwalurvi22@gmail.com

Experiences

WEB DEVELOPMENT INTERNSHIP AT SDG INDIA PVT. LTD.

MAY 2019 - JULY 2019

- Worked on the AEO Project for Dubai Customs. Developed web pages using HTML, JavaScript and jQuery

Education

B. TECH | 2016-PRESENT | JIIT, NOIDA

- Branch: CSE Integrated
- Current GPA: 8.2/10

HIGHER SECONDARY | 2014-2016 | R.K.P.S, AMROHA

- Subjects: Physics, Chemistry, Mathematics
- Percentage: 88%

HIGH SCHOOL | 2012-2014 | ST. MARY'S CONVENT SCHOOL, AMROHA

- GPA: 10/10

Skills & Abilities

TECHNICAL

- C/C++ programming
- Web Development (HTML, JavaScript, PHP, MySQL, Node.js)
- Worked on Arduino-based projects
- Python programming

OTHER

- Good at mathematics and reasoning.
- Team work spirit.

Projects

METRO ROUTE FINDER:

- A c++ based project to find shortest route in the Delhi metro and calculate minimum cost, minimum time, number of switches required that would ease up the commute for new as well as existing travelers.

DROUGHT PREDICTION:

- A program developed using Arduino that could predict a drought comparing present stats with some pre-fed data. Useful for farmers. Used gps module, soil humidity and temperature sensors.

LIBRARY MANAGEMENT:

- A website created using HTML, JavaScript, PHP on which information of certain books was saved using MySql database.
- The user could search through the books applying various filters.

BLOCKCHAIN IMPLEMENTATION:

- The blockchain model was implemented using Node.js and servers were run with the help of Postman. Certain tweaks in the architecture were also introduced.

SEMANTIC SEARCH ENGINE FOR LOCAL DESKTOP:

A query in a given document pool will result in semantically related terms. Implemented using techniques like Keyword Extraction and Word Embedding. This is an NLP based project.

Research Work

DISCOVERING MOTIFS IN DNA SEQUENCES: A SUFFIX TREE BASED APPROACH

- Developed a suffix tree and trie based algorithm for the motif finding problem in bioinformatics.
- Significant optimization in the pre-processing time.

CPDP: A CONNECTION BASED PDP ALGORITHM

- Worked on the partial digest problem, also known as turnpike problem and developed a python-based program to find the solutions.
- Significant optimization as compared to the present algorithms.

PARTIAL DIGEST PROBLEM

- A chapter in the book Smart Healthcare Systems (Taylor & Francis).
- Comprehensive explanation of the Partial Digest Problem along with description major algorithms with their respective codes.

NAYAN GARG

Mobile number: 7985072692
Email Address: nayan21garg@gmail.com

Profile

Objective Intend to do an intern with leading corporations in the field of Technology and work alongside dedicated, like-minded people, which will help me, explore myself fully and realize my true potential. Willing to work on innovative and challenging projects that can shape the future of the world.

Education

2016-2021	**Integrated B.Tech and M.Tech (CSE)** Jaypee Institute Of Information Technology	**7.4 (after 6 sem)**
2010	**Senior Secondary Examination - Class XII** Boys' High School, Allahabad ISC	**92%**
2008	**Higher Secondary Examination - Class X** Boys' High School, Allahabad ICSE	**93.8%**

Projects

- **RESEARCH PAPER ON SORTING BY REVERSALS**
  - Overlap Forests is a data structure, which is widely used for solving problems related to "Sorting By Reversal" in molecular biology.
  - We worked on optimizing the construction of Overlap forests, and were able to reduce the construction time by 1%(in the worst case) to 83%(in the best case).
  - All computations were done on standard Microbiology datasets.
  - We wrote a research paper, that was accepted in an IEEE Conference(SPIN-2019) and presented in March 2019
  - Technologies used - Python
- **GVA CALCULATION SOFTWARE**
  - GVA (Gross Value Added) is metric which is derived from audit data of various industries. This gives the gross value added by specific industries to the national economy, which makes it a metric of national importance.
  - Our software automated the process of compiling Audit data from different industries, and reduced the manual effort involved in the process.
  - This project won the second prize in the Smart India Hackathon 2019 (a Government of India initiative) for the Ministry of Statistics and Program Implementation.
  - Technologies used - Python, Highcharts APIs, MySql.
- **MUSI-KING:-**
  - A C++ based music player where one could listen to his favourite songs. The user could search for songs, sort the songs, create his own playlist, add songs to his playlist, delete the songs, shuffle them, search autocomplete etc.
  - We used data structures like TRIE(prefix tree), Binary Search tree and Hash Tables, to

speed up the search and provide a better user experience.
    - Technologies Used - C++

- **AUTOMATED RESTAURANT:-**
    - A working model of a digital age restaurant where the customers do not have to wait for the waiters to place their orders.
    - A map of the restaurant outside would tell the customer where to find an empty table.
    - Now, when he sits on the table and picks up his menu card he is detected using ultrasonic sensors. While remaining at his seat he can place orders using a keypad placed there. The order would go directly to the chef who would update the time remaining for their order on a common LCD.
    - We used an ARDUINO MEGA board equipped with ATMEGA 2560 microcontroller. Bluetooth module HC-05 was used for communication.
    - This was presented in the International Conference of Contemporary Computing, 2017, exhibition and gained large interests from the attendees.

- **GESTURE CONTROLLED ROBOT :-**
    - We developed a robot, that could be controlled by human gestures with the help of a glove.
    - We used accelerometer, NRF module transmitter on the glove to capture and transmit the gestures to the bot.
    - The bot was fitted with an ATMEGA-328 coupled with an NRF receiver, to receive the signals, decode them and act accordingly.

- **MY FUTURE WEBSITE**:-
    - As one of my first college projects, I developed a website that could help students to decide their prospective Engineering colleges, based on their rank in entrance exams.

- **MAZE SOLVING ROBOT :-**
    - A robot to solve a 2-D maze implemented on Atmega-328.

Experiences

- Summer Internship at Internity in Android Development.
- Summer Internship at IIIT Allahabad in Machine Learning.

Accomplishments

- Runners-up, Smart India Hackathon-2019 (A Government Initiative)
- Presented a research paper (Sorting by Reversals: A faster Approach for Building Overlap Forest) at an IEEE Conference(SPIN-2019)
- 2nd prize among 1st years, Execute 16.2, a competitive programming event at JIIT-62.
- Organized multiple events for Microcontroller Based Systems and Robotics Hub and IEEE Student Branch during 2nd Year of College
- 2nd prize at 4 Robotics based events held in JIIT.
- 3rd prize at a Robotics based event held in JIIT.
- My best rank in Codechef Long Challenge has been under 600.

Skill Set and Interests

- **Programming Languages:-**
  - Proficient:- C,C++
  - Average:- Python
  - Introductory:- Android, Java, PHP, Matlab, JavaScript, MySQL, NodeJs, Octave.
- **Operating Systems:-**
  - Windows
  - Linux

Interests:-

- Machine Learning
- Algorithm Design
- Competitive Programming
- Robotics

Personal Details

- **Date of birth :** 28 March, 1998
- **Alternate Number:** 9560276575
- **Nationality :** Indian
- **Permanent Address:** 144, Alopibagh, Allahabad, U.P.
- **Current Address:** Jaypee Institute Of Information Technology, A-10, sector-62, Noida, U.P.

SUGANDHA SINGHANIA

8860637400 | sugandha.singhania36@gmail.com

EDUCATION

Integrated B.TECH - M.TECH | 2016-2021 | JIIT, NOIDA

- Branch: CSE (Dual-Degree), 3rd year
- Current CGPA: 7.3/10 (after 6th semester)

HIGHER SECONDARY | 2014-2016 | ST. THOMAS SCHOOL, INDIRAPURAM

- Subjects: Physics, Chemistry, Mathematics
- Percentage: 87.8%

HIGH SCHOOL | 2012-2014 | ST. THOMAS SCHOOL, INDIRAPURAM

- GPA: 10/10

PROJECTS

- **MUSI-KING**

  A C++ based music player where one could listen to his favourite songs. The user could search for songs, sort the songs, create his own playlist, add songs to his playlist, delete the songs, shuffle them, etc.
  While searching, the user gets suggestions of the song they might be searching for. For this, the song names were stored in a TRIE(prefix tree).
  For easier searching and sorting the application used Binary Search Trees and hash tables.
  New songs could be added directly to the file system of the PC as the project built its library from there itself. So any changes to the folder would be reflected in the song library

- **MY FUTURE**

  A website where students can decide their future college after Entrance Exams based on their preferences. They could search colleges based on their rank and other criterions. They could get entire details about the college right there.

- **BLOCKCHAIN IMPLEMENTED:**

  Big data handling using blockchain technology. Implemented the blockchain architecture in Node.js with certain modifications. Used Postman for running dummy servers.

RESEARCH WORK

SORTING BY REVERSAL PROBLEM:

The research paper has been accepted in an IEEE Conference (SPIN-2019) and published in March 2019.

- Developed a breakpoint graph and overlap forest based single scan algorithm for the sorting by reversals problem in the bioinformatics.
- Significant optimization in time complexity as compared to present algorithm

EXPERIENCES

- Did summer internship at iONE IT Solutions Pvt. Ltd. in web development.

SKILLS & ABILITIES

TECHNICAL

- C/C++ programming
- Web Development (HTML, JavaScript, PHP, MySQL)
- Python programming

OTHER

- Good at mathematics and reasoning.
- Team player
- Good at painting and drawing

OTHER ACHIEVEMENTS

- Hosted multiple college events throughout 1st and 2nd year including Fresher's 2017, Farewell 2018, etc.
- Organised technical as well non-technical events such as Cyber Sristhi(tech fest of JIIT), Impressions(cultural Fest of JIIT), Alumni Meet, etc. as Organising Committee member.
- Was part of Start up as a marketing intern.

PERSONAL DETAILS

- Date of Birth : 10th September 1997
- Permanent Address : 2074, Pragya Kunj, Sector-4C, Vasundhara, Ghaziabad, Uttar Pradesh.