# Efficient Similarity Search in Web Player

M.Nandhini
UG Scholar
nandhinimallaiyan@gmail.com
Department of computer science and engineering
IFET college of engineering
P.Divya
Assistant Professor
divi.diviner@gmail.com
Department of computer science and engineering
IFET college of engineering

**Abstract: Similarity search is promising criteria in data mining technique which returns matching data . To improve the similarity search, The proposed to use top-k query matching algorithm which involves getting multiple input from the user and produce better similarity search over the data. According to our algorithm it gives three kind of results based on similarity such as exactly matched, almost matched and moderately matched to the typed query. Moreover we can view any kind of data through the application such as audio, video ,pdf and other text format data's. And more important it gives suggestions based on similarity. This project's some other skin texture and rest of the portions are explained in below contents.**

**KEYWORD: Similarity,Query,Index.**

## INTRODUCTION

In any field we store data's and view accordingly is a major process in real time applications. We search stored document based on the needs lot of algorithms proposed to solve the similarity search issues. In this top-k similarity search is one of them mostly used for similarity search. In our proposed concept we used this algorithm by classification of exactly matched content, almost matched content and moderately matched content. We used this top-k similarity search in e-learning concept toview all kind of data files such as audio,video, docs,text and pdf files. More overwe can play and view these files by using our application. Top-k similarity search algorithm uses multiple keyword input from a user and search the database with these given details. The processing technique involves multipleinputandmultiple output. This algorithm matches the input with keyword and produce similar contents according its matched rate. It will be defines searched data into three category 1. Exactly matched content 2.Almost matched content 3.Partially matched content. This application allows user to view notification for newly added content. It has two category 'today updates' and 'this week updates'. We can directly view this data through the link. And we have a player to play the uploaded files.

## LITERATURE SURVEY

### [1]Locality-sensitive hashing scheme based on p-stable distributions

A novel Locality-Sensitive Hashing scheme for the Rough Nearest Neighbours Problem under $l_p$ standard, based on *p*-stable deliveries.Our scheme expands the running time of the earlier algorithm for the case of the $l_p$ standard. It also yields the first known provably efficient rough NN algorithm for the case $p<1$. We also show that the algorithm finds the exact near neighbour in $O(\log n)$ time for data satisfying certain "restricted growth" condition.Unlike earlier schemes, our LSH scheme works directly on points in the Euclidean space without embeddings. Accordingly, the resulting query time bound is free of large issues and is simple and easy to implement. Our experiments (on synthetic data sets) show that the our data structure is up to 40 times faster than *kd*-tree.

### [2] Learning to hash with binary reconstructive embeddings

Fast retrieval methods are increasingly critical for many large-scale enquiry tasks, and there have been several current methods that attempt to learn hash functions for fast and exact nearest neighbor searches. We develop an algorithm for learning hash functions based on explicitly reducing the rebuilding error between the original distances and the Ploy distances of the consistent binary embeddings. We develop a scalable coordinate-descent algorithm for our proposed hashing objective that is able to efficiently learn hash functions in a variety of settings. Unlike existing methods such as semantic hashing and spectral hashing, our method is easily kernelized and does not require restrictive assumptions about the underlying delivery of the data. We present results over several domains to validate that our method outclasses existing state-of-the-art techniques.

 

**[3] Minimal loss hashing for compact binary codes**

A method for learning similarity protective hash functions that map high dimensional data onto binary codes. The formulation is based on structured estimate with latent variables and a hinge-like loss function. It is resourceful to train for large datasets, scales well to large code lengths, and outclasses state-of-the-art techniques.

## SYSTEM MODEL:

***Indexed Meta Data***

Generally we use indexing method for any one of criteria(column) for searching purpose .But in this module we make no of indexed column for efficient searching and to get a better similarity. In this module contains only meta data of stored files.

***Admin***

The admin will be uploading the files from the data user. He can upload all format files such as video , audio , pdf , doc and etc. Uploading process should be encrypted by "multipart/form-data" . This file received by using cos_multipart.jar file. When uploading files the administrator should mention the category , topic and subtopic for efficient searching purpose.The admin will produce question and answer paperand also exam time and date are considered.

***MIQ(Multiple Input Query)***

MIQ which gets number of input from the user and search for the similarity by using top-k matching algorithm. This is not a necessary to give no of inputs. But should give atleast one input for searching process. For the initial stage we can give maximum three inputs.

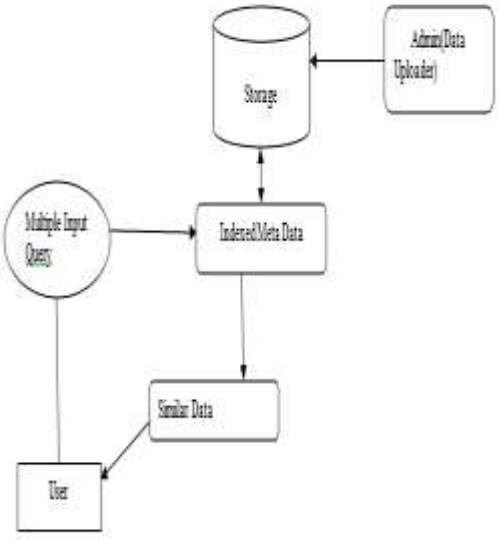


**Fig 1:System Architecture**

**Top-k similarity search**

This exact-match query model is not suitable for many database applications and scenarios where queries are naturally fuzzy often expressing user preferences and not durable Boolean constraints and are best answered with a ranked list of the best similar objects, for some definition of quantity of match. Sometimes the "attributes" of the data objects over which a top-*k* query is issued are handled by external, self-directed databases existing in excess of the web.


**Fig 2:Similarity search**

**Player**

In this module plays the media files such as audio and video format . Also we can download and view these all .Play will shows the current file which we selected from the list of contents. Play shows the current file which we select from the list of contents. This suggest exactly matched with the current file that in play are area. Almost matched files are allocated for by the top right corner.Minimum similarity files will be produced by partially matched.

## ALGORITHM

**Input:** data set $D$, int $k$ (the number of similar pairs to search for), int $m$ (hash vector length), int $t$ (the number of hash tables), int $w$ (the maximum hamming distance between buckets)
**Output:** $kPSet$ (the sorted list of top $k$ pairs in decreasing order of JoinSim)

Build advanced LSH indexing for $D$ according to Section 4.2;
Initialize $kPSet$ to $\emptyset$ and pruning similarity $z$ to -1;
**for** each hash table $T_{H_m^i}$ $(1 \leq i \leq t)$ **do**
  **for** each bucket $B$ **do**
    Generate all possible node pairs from $B$ and add them to $kPSet$ if their JoinSim is larger than $z$;
  **end for**
  **for** each bucket pair $(B_s, B_{s'})$, $s \neq s'$ **do**
    **if** the hamming distance between $B_s$ and $B_{s'} \leq w$ **then**
      Check each mismatch bit in the hash vectors of $B_s$ and $B_{s'}$ and prune $(B_s, B_{s'})$ once it satisfies the pruning strategy at a mismatch bit;
      **if** $(B_s, B_{s'})$ is not pruned **then**
        Generate all possible node pairs from $B_s$ and $B_{s'}$ and add them to $kPSet$ if their JoinSim is larger than $z$;
      **end if**
    **end if**
  **end for**
**end for**
**return** $kPSet$;

 

## CONCLUSION

Based on goal of our project we successfully demonstrated multi input query processing with top-k mining similarity search. The evaluation result of our project gave a considerable output than existing systems. And the Exactly matched , almost matched and partially matched content concept gave good result to typed keyword. We done our project for educational purpose to search ,view notes and write exams through online. More over the question paper format designed to support all format such as audio, video and text .Also this can be apply to the notes . In future we can provide some enhances in online exam writing .In writing exam portion there are three types like choose, fill-up and paragraph. We should create efficient content matching in fill-up and paragraph type.

**REFERENCES**

[1]G. Navarro and N. Reyes, "Dynamic list of clusters in secondary memory," in SISAP, pp. 94–105, 2014.
[2] J. Almeida, R. D. S. Torres, and N. J. Leite, "BP-tree: An efficient index for similarity search in high-dimensional metric spaces," in CIKM, pp. 1365–1368, 2010. [3] G. Ruiz, F. Santoyo, E. Chavez, K. Figueroa, and E. S. Tellez, "Extreme pivots for faster metric indexes," in SISAP, pp. 115–126, 2013.
[4] J. Lokoc, J. Mosko, P. Cech, and T. Skopal, "On indexing metric spaces using cut-regions," Inf. Syst., vol. 43, pp. 1–19, 2014.
[5] D. Novak, M. Batko, and P. Zezula, "Metric Index: An efficient and scalable solution for precise and approximate similarity search," Inf. Syst., vol. 36, no. 4, pp. 721–733, 2011.
6. C. T. Jr., R. F. S. Filho, A. J. M. Traina, M. R. Vieira, and C. Faloutsos,"The Omni-family of all-purpose access methods: A simple and effectiveway to make similarity search more efficient," *VLDB J,*2007.
7. L. Mico, J. Oncina, and R. C. Carrasco, "A fast branch & bound nearest neighbour classifier in metric spaces," *Pattern Recognition Letters*,1996.
8. P. N. Yianilos, "Data structures and algorithms for nearest neighborsearch in general metric spaces," in *SODA*, 1993.
9. E. Vidal, "An algorithm for finding nearest neighbors in (approximately)constant average time," *Pattern Recognition Letters*,1986.
10. L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen, "Efficient metricindexing for similarity search," in *ICDE*, 2015.

 