International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 08 | Aug -2017 www.irjet.net p-ISSN: 2395-0072

PROJECTION MULTI SCALE HASHING KEYWORD SEARCH IN MULTIDIMENSIONAL DATASETS

***1 Mrs. S. Kalaiarasi , *2 Mrs. S.A.Shoba**

**1 M.Phil Research Scholar, PG & Research Department of Computer Science & Information Technology, Arcot Sri Mahalakshmi Women's College, Villapakkam, Tamil Nadu, India.*
**2. Head of the Department, PG & Research Department of Computer Science & Information Technology, Arcot Sri Mahalakshmi Women's College, Villapakkam, Tamil Nadu, India.*

Abstract: Keyword-based search in text-rich multi-dimensional datasets facilitates many novel applications and tools. In this paper, we consider objects that are tagged with keywords and are embedded in a vector space. For these datasets, we study queries that ask for the tightest groups of points satisfying a given set of keywords. We propose a novel method called ProMiSH (Projection and Multi Scale Hashing) that uses random projection and hash-based index structures, and achieves high scalability and speedup. We present an exact and an approximate version of the algorithm. Our empirical studies, both on real and synthetic datasets, show that ProMiSH has a speedup of more than four orders over state-of-the-art tree-based techniques. Our scalability tests on datasets of sizes up to 10 million and dimensions up to 100 for queries having up to 9 keywords show that ProMiSH scales linearly with the dataset size, the dataset dimension, the query size, and the result

Keywords: ***Keyword search Engine, ProMish, Novel method, Tree based Techniques.***

I. INTRODUCTION

The last two decades have witnessed exponential growth of the World Wide Web (WWW). On the Internet, people have access to an almost infinite amount of diverse information and content. The popularity of the Internet, which has made information available to users on an unprecedented scale, has led to a new information age.

The growth of the Internet has led to an explosion of information. There are too many data and sources for users to deal with in such a way as to locate the information that is most relevant to them. This phenomenon is called the information overload problem. For example, it is difficult for us to make choices from thousands of movies and music, millions of books, billions of web pages and so on. Indeed, to evaluate these items one by one is an impossible task. In order to help online users cope with large amounts of diverse data on the Internet, intelligent software agents have emerged with the purpose of enabling users to find the relevant items that meet their needs. Motivated by this, the research on web personalization is rapidly developing. Personalization is the ability to provide content and services tailored to individuals based on knowledge about their preferences and behavior User profiling is the process of collecting user data and creating computerised user profiles that represent specific users' personal interests and needs. User profiling is the very foundation of web personalization. Therefore, the accuracy of user profiles directly affects the quality of web personalization. While Internet search engines, such as Google and Bing, are an essential tool for helping users to get their desired information, they do not take into consideration the users' personalized interests and preferences, and return the same results to the queries of users with different needs**.**

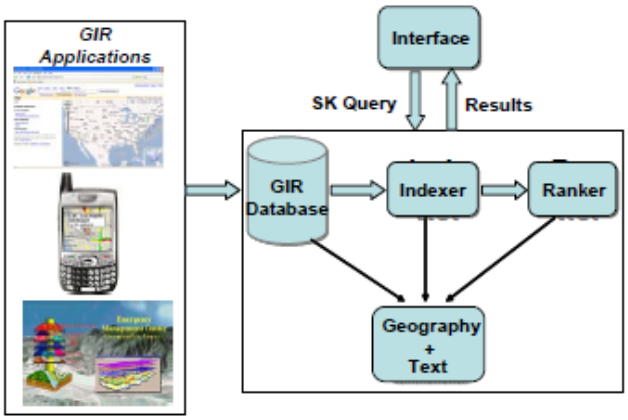


Fig.1.1 GIR System Architecture

GIR Applications: Before describing the components of the framework, we briefly explain different GIR applications that can fit within the framework. The class of applications that take as input SK queries, process them and output information sources that are relevant to the queries are called as GIR applications. Geographic (or local) search engines, location-based services (LBS) and GIS search engines are various examples of GIR applications. Geographic search engines treat Web pages as information sources and process SK queries on top of them. GIS search engines process SK queries on top of GIS databases.

Objective 1: develop a multifaceted model representation for the multidimensional social tagging data. In multidimensional data like social tagging data, users, items and tags are all linked one to another. Users collect certain items with certain tags; items are collected by some users using certain tags; tags are used by users to annotate some items. Thus, first of all, it is essential to employ an efficient method of representation to model the data without losing the multidimensional relations between all dimensions within the data.

Objective 2: develop user/item profiling techniques with high-order relations based on the multidimensional social tagging data. As we previously discussed, the traditional user/item profiling techniques discard much information due to the dimension projection strategy; thus, it is critical to preserve the high-order information within the multidimensional data in the profiles, if more accurate user/item profiles are desired.

Objective 3: integrate the proposed user/item profiles into recommender systems. In order to verify the effectiveness of the proposed user/item profiling approaches, user/item profiles generated based on the proposed profiling techniques will be integrated with several collaborative filtering recommendation systems. The recommender systems will be investigated and evaluated in extensive experimental studies.

II. RELATIVE WORK

Previous works on profile-based PWS mainly focus on improving the search utility. The basic idea of these works is to tailor the search results by referring to, often implicitly, a user profile that reveals an individual information goal. In the remainder of this section, we review the previous solutions to PWS on two aspects, namely the representation of profiles, and the measure of the effectiveness of personalization.

Many profile representations are available in the literature to facilitate different personalization strategies. Earlier techniques utilize term lists/vectors or bag of words to represent their profile. However, most recent works build profiles in hierarchical structures due to their stronger descriptive ability, better scalability, and higher access efficiency. The majority of the hierarchical representations are constructed with existing weighted topic hierarchy/graph, such as and so on. Another work in builds the hierarchical profile automatically via term-frequency analysis on the user data. In our proposed UPS framework, we do not focus on the implementation of the user profiles. Actually, our framework can potentially adopt any hierarchical representation based on taxonomy of knowledge.

The above problems are addressed in our UPS (literally for User customizable Privacy-preserving Search) framework. The framework assumes that the queries do not contain any sensitive information, and aims at protecting the privacy in individual user profiles while retaining their usefulness for PWS.

- When a user issues a query q on the client, the proxy generates a user profile in runtime in the light of query terms. The output of this step is a generalized user profile G satisfying the privacy requirements. The generalization process is guided by considering two conflicting metrics, namely the personalization utility and the privacy risk, both defined for user profiles.
- Subsequently, the query and the generalized user profile are sent together to the PWS server for personalized search.
- The search results are personalized with the profile and delivered back to the query proxy.
- Finally, the proxy either presents the raw results to the user, or re-ranks them with the complete user profile.

Recommender systems are used as an efficient tool for dealing with the information overload problem. Based on how recommendations are made, recommender systems are usually classified into the following categories: content-based recommendations, collaborative recommendations and hybrid approaches Collaborative filtering (CF) is the most widely used technique for making recommendations. A large number of CF recommendation approaches have been developed in recent years. For example, user-based and item-based CF are two important neighbourhood-based CF approaches, which are directly linked to k-nearest neighbours algorithms (k-NN). Latent factor models or Keyword Search models are another important family in CF which gains much attention due to its competitive performance. For the traditional recommendation problem based on a two-dimensional user-item matrix, both neighborhood-based CF and latent factor models have been accepted as effective recommendation approaches in theory and in practice

2.1 Notation

For the purposes of this work, the following terminology is used. We suppose a set of objects X with m attributes Ai,...,A_{TO}. Each attribute A has its own *attribute domain,* which we denote D_{Ai}. The attribute domain D_{Ai} is a set of relevant values allowed for the attribute Aj. Every object x *G* X has m attribute values A_1 (x), A_m(x) from domains D_{Al},D_{Am} of attributes A_1,...,A_m, respectively, i.e., Aj(x) G D_{Ai} for i= 1,...,m. We differentiate between the attribute domain and an *actual attribute domain.* The actual attribute domain D^f is the set of values Aj(x) of all objects x G X. Consequently, the relationship C D_{Am} holds.

Ranking function

In the top-k search, the user's preferences are modeled by a ranking function, which assigns a ranking for each object from a data set. In some of the related works the ranking is denoted as a score. According to the ranking function, it is possible to sort objects from the data set and to find the best k objects according to the user preferences. More formally, ranking function R evaluates the ranking (overall score) of each object x G X according to all its attribute values A1 (x), Am(x) and the value of ranking is R(A1(x),Am(x)). The ranking function R maps every object x G X into the real numbers, i.e.,

$$R(_{ai,am}) : D\text{^}i\ X \ldots x\ D_{Am} \text{ ->= } R.$$

For better clarity, we can restrict the range of values of the ranking func¬tion on the unit interval [0,1], where 0 means no preference and 1 means the highest preference. In the top-k search, this restriction is a very common convention. In short notation, we denote a ranking of the object x as a function R(x) = R(Ai(x),Am(x)) with one variable, which we understand as a mapping

$$R(x) : X \text{ — } [0,1].$$

2.2 Fuzzy logic perspective

Our model of user preferences is motivated by fuzzy logic. In classical two-valued logic it would be possible to represent only whether the object *x £ X* is suitable (true value, ranking is 1) or not (false value, ranking is 0). In that case, the range of the ranking function is set {0, 1}.

Multi-valued logic, namely fuzzy logic, may have more than two truth values. This may be expressed as existence of various degrees of truth. In fuzzy logic, the set of truth values is the unit interval of real numbers [0, 1]. In context of user preferences, the degree of truth, which attains values in the interval [0, 1], is interpreted as the preference, where 1 means the most preferred and 0 the least preferred. In our model, we express a local preference for ith attribute A_i as a fuzzy function f_i, which can be defined in the same way as the local preference function, i.e.,

$$f_i(Ai(x)): D_{Af} \text{- } [0,1].$$

III. PREVIOUS IMPLEMENTATIONS

Nearest neighbors search, also known as similarity search is a search for the closest point in metric spaces. We show that the *k-nearest neighbors* (k-NN) search is a special case of the top-k search. For this purpose, we can perceive a Cartesian m-dimensional product space D over domains of attributes A }, A_m as Euclidean space and use Euclidean distance as the metric on this space. Supposing the set of points D^X we can find the nearest (or the most similar) k objects in D^X to a given query point.

The Euclidean distance between points p = (p },...,p_m) and q = (q },...,q_m) is defined as the length of the line segment connecting these two points. The Euclidean distance d(p, q) is given by

$$d(p, q) = \sqrt{(pi - qi)^2 + \ldots + (pm - qm)^2}.$$

The Euclidean distance is also perceived as a measure of similarity between two points. In the k-NN search, we are searching for the k points with the smallest distance from a query point q. But when using the model, we are searching for the best k objects with the highest value of a ranking function. This is possible to solve by a transformation of the Euclidean distance d from a given query object q into a ranking function R. This ranking function has to express for each object x £ X its value of $R^u(x)$ dependent on the Euclidean distance of x (or similarity) from the given query object q, i.e. d(x, q).

Moreover, when we compare R(x) in the top-k search to d(x, q) in the k-NN search, the best object x has R(x) almost equal to 1 instead of d(x, q) almost equal to 0 and the best object x has R(x) almost equal to 0 instead of d(x, q) almost equal to oo. This problem can be solved by normalization of the distance into unit interval [0,1] and then by subtracting the normalized distance from 1.

Finally, when using this model, the transformation of the k-NN search to the top-k search is based on the composition of local and global preferences. We use local preference function

Algorithm 1 TA algorithm (Fagin's threshold algorithm)

1.TA(sorted lists L\, ...,Lm, aggergate function g, number k) top-k := 0;
2. for 1 < i < m do highi := 1; 3: threshold := 1; 4: min-k := 0; 5: repeat
3. for 1 < i < m do
4. {x,Ai(x)} := GETNEXTPAiR(Li); // sorted access
5. high := Ai(x);
6. threshold := g(high\, ...,highm);
7. for 1 < j < m do
8. if j = i then Aj(x) := GETVALUE(LJ, x);
9. end for
10. score(x) := g(Ai(x),Am(x));
11. if score(x) > min-k then
12. if top-k.SIZE() = k then top-k.REMOVEKTHOBJECT();
13. top-k.INSERTINTHERIGHTPLACE(X);
14. min-k := top-k.GETMIN-K();
15. end if
16. end for
17. until threshold < min-k 21: return top-k;

Provides an estimation of the highest possible attribute value, which can be obtained by performing of the next sorted accesses in list Li. In this way, it is possible to estimate attribute value Ai(x) of each object x, which has not yet been seen as a pair {x, Ai(x)} in list Li. Based on these estimations, the threshold value is evaluated by aggregate function g, which combines all the last seen attribute values in the lists, i.e., threshold = g(high\,highm). Since the aggregate function has to be monotone and the lists are sorted in the descending order, the TA algorithm can estimate score of each object, which has not yet been seen in any of the lists. It means that score of such that object can be only equal or lower than is the threshold value.

	L_2	L_3	
K, 1.0}	K 1-0}	{*„ 1.0}	*i
{x_2, 0.8}	{x_4, 0.8}	{x_4, 0.8}	x_2
JCJ, 0.6}	{x_6, 0.6}	{x_3, 0.6}	X,
K, 0.4}	K 0.4}	{x_5, 0.4}	x
			A
{x_5, 0.2}	{x_2, 0.2}	{x_2, 0.2}	x_5
{x_6, 0.0}	{x_5, 0.0}	{x_6, 0.0}	x_6

Each new object x acquired by the sorted access is obtained with value Aj(x) of only one attribute Aj depending on the list Lj, from which it was obtained. Because the NRA algorithm does not use the random acces, any other attribute values of the object x, except the value Aj(x), can be obtained only by sorted access from the other sorted lists different from list Lj. It is not possible immediately, it can occur in one of the further sorted accesses performed in the other lists, i.e., if the NRA algorithm obtains a pair {x,Aj(x)} from list Lj different from list Lj.

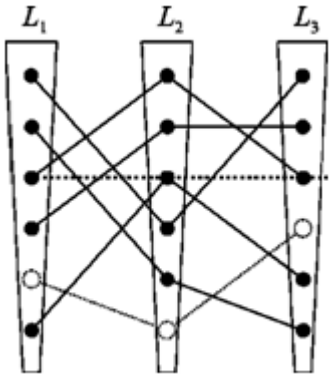


Fig .3.1 New Object detection

IV. PROPOSED ANALYSIS

In our research, user, item and tag are three fundamental entities of data. For the three basic entities, we define {/, / and T as disjoint non-empty finite sets. We use set symbols in lower case with indices in the subscripts to denote individual elements of a set, for example, u_2 denotes the second user in the user set {/. We use symbols of individual set elements, i.e., user, item iy, or tag t_p, in the subscripts or superscripts of the set symbols, as conditions imposed onto the sets to denote subsets. For example, denotes the item set in which each item was used by the vth user u„ 6 U.

- Users. U = u_2, — , u^, — ,U|y|} contains all users in a social tagging system. denotes the vth user, $u_5$6 0, 1 <v < |a[a/b|.
- Items. / = (t_1, t_2, —, ty, —, t|$_7$|} contains all items used by the users in the system. iy denotes the y'th item, iy 6 /, 1 < _/'< |a/b|.
- Tags. T = (t_1, t_2, —, t_p, —, t|$_T$|} contains all tags used by the users in the system. t_p denotes the pth tag, t_p 6 T, 1 < p < |T|.
- Tag assignments. The basic tagging behaviour, namely tag assignment, is defined as a 3-tuple e: U x / x T 6 {0,1}. If a user collected item t; with tag t„, then e„ $_t$ =1, otherwise e„ $_t$ =0.

4.1 User Profiling Based On Multidimensional Singular Value Decomposition

Traditionally some two-dimensional CF approaches apply SVD on a user-item matrix to compute user or item profiles and identify similar users/items (Symeonidis et al., 2006; Mi Zhang & Hurley, 2009). In these approaches, user-item matrix M 6 R't/'x'7' is decomposed and approximated by the truncated SVD:

Taking user-based CF, for example, <W|[/|XFC 1 £fcxfc is used to project each user's data from an |/|-dimensional space to a k-dimensional space, where k principal components of the data are preserved. Thus the user profile matrix in the SVD-based user profiling is computed as:

4.2 Incorporating Nearest Neighborhood

Top-N item recommendation tasks in the multidimensional data context have received increasing attention in the last decade. Tensor factorization models and neighbourhood-based collaborative filtering are two major techniques in use. They address the item recommendation task in quite different ways and also have different strengths. In this chapter, we propose two novel collaborative filtering multidimensional recommendation approaches for top-N item

recommendation tasks in the context of social tagging systems. The first one is neighbourhood-enhanced tensor factorization collaborative filtering (NTF). NTF conducts ternary latent semantic analysis, and then updates the tensor factorization using k-NN methods in terms of the users with similar interests. The second one is tensor factorization for neighbourhood-based collaborative filtering (TFN).

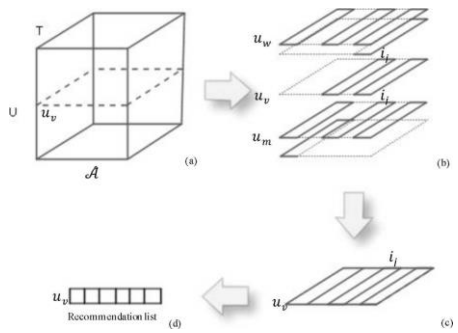


Fig .4.1 NTF recommendation making for user

4.3 Nearest Search Using Semantic Web Technology

In this research here have proposed Ontology based E Learning Management System where basic tools administration, Instructor, Learner are interrelated through Learning Resource (RDF) and Ontology-based Contextual Knowledge (OWL). Where each tool contains several elements that are given in the figure-1.This model is designed with six subsections. That are marked (1, 2, 3..6) in our Proposed ontology based E-LMS model

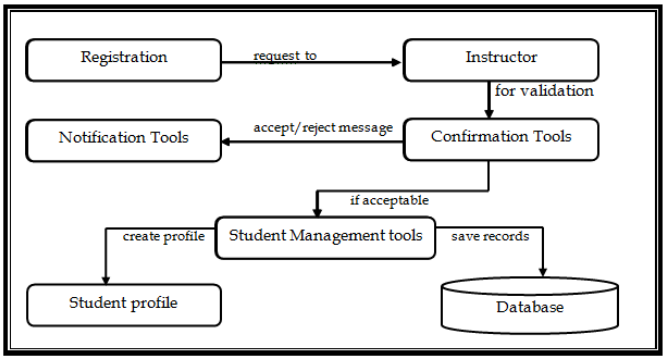


Fig .4.2 Nearest Search using Semantic web Technology

To illustrate the overall procedure, we will go through an e-learning scenario. A student first search for an online course: the broker handles the request and returns a set of choices satisfying the query. If no course is found, the user can register with a notification service. Otherwise, the user may find a suitable course among the offerings and then makes a final decision about registering for the course. Processing the registration can be seen as a complex service involving registering with the system, creating a confirmation notification, creating a student account (authentication/ authorization), and providing learning materials.

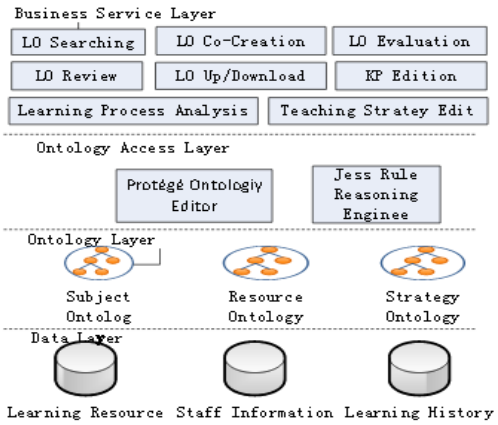


Fig .4.3 Ecological system model

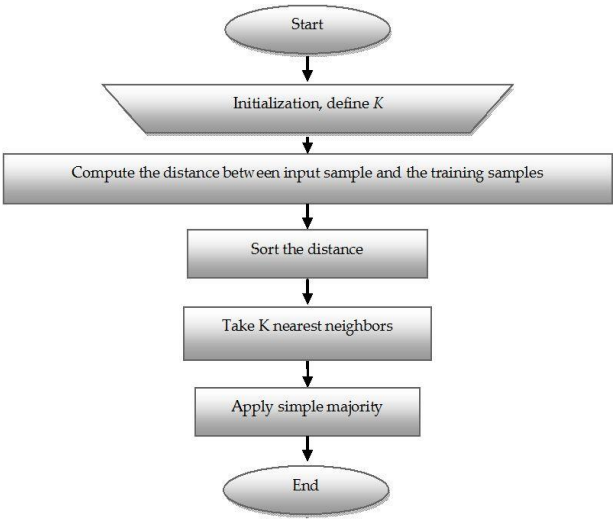


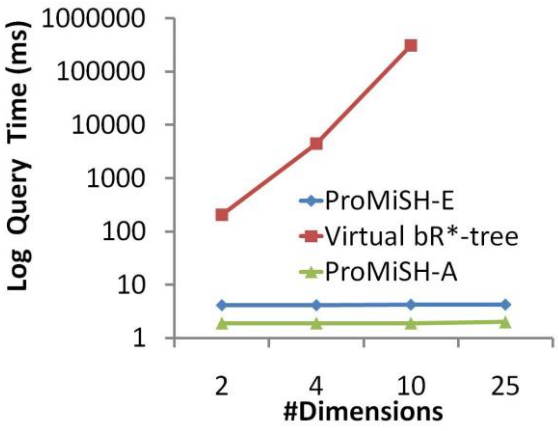
Fig 4.4 Flow Chart Naive Bayes

Given that the concepts and click through data are collected from past search activities, user's preference can be learned. These search preferences, inform of a set of feature vectors, are to be submitted along with future queries to the PWS server for search result re-ranking. Instead of transmitting all the detailed personal preference information to the server, PWS allows the users to control the amount of personal information exposed. In this section, we first review a preference mining algorithms, namely SpyNB Method that we adopt in PWS, and then discuss how PWS preserves user privacy. SpyNB learns user behavior models from preferences extracted from click through data.

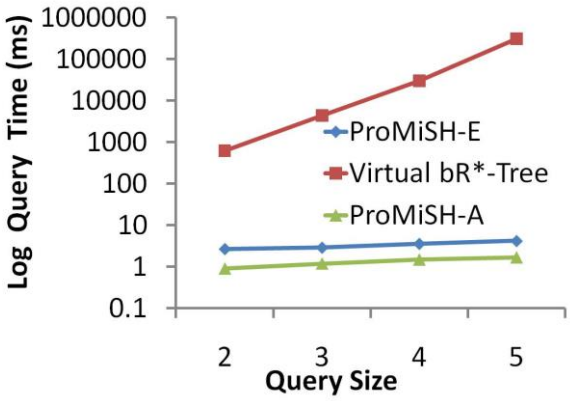
V. EVALUATION RESULT

Here evaluated the performance of ProMiSH-E and ProMiSH- A on synthetic and real datasets. We used recently introduced Virtual bR*-Tree [2] as a reference method for comparison (see section II for a description). We first introduce the datasets and the metrics used for measuring the performance of the algorithms. Then, we discuss the quality results of the algorithms on real datasets. Next, we describe comparative results of ProMiSH-E, ProMiSH-A, and Virtual bR*-Tree on both synthetic and real datasets. We also report scalability results of ProMiSH on both synthetic and real datasets. Finally, we present a comparison of the space usage of all the algorithms.

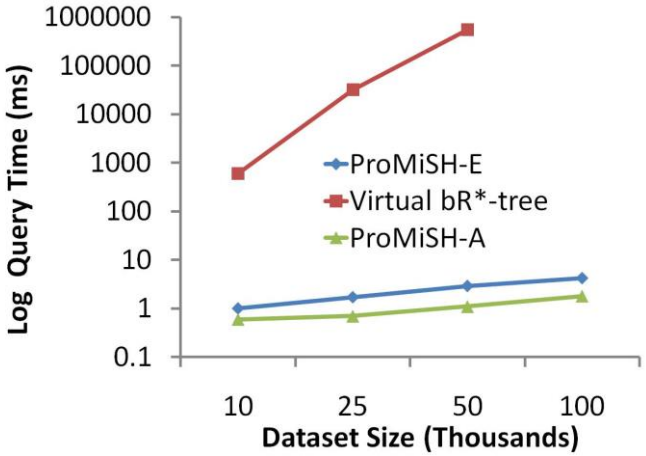
Datasets: We used both synthetic and real datasets for Experiments. Synthetic data was randomly generated. Each component of a d-dimensional synthetic point was chosen uniformly from [0-10,000]. Each synthetic point was randomly tagged with t keywords. A dataset is characterized by its (1) size, N; (2) dimensionality, d; (3) dictionary size, U; and (4) the number of keywords associated with each point, t. We created various synthetic datasets by varying these parameters for our empirical studies.



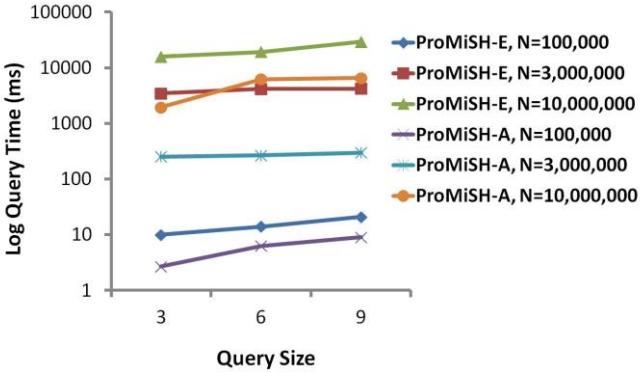
Query time comparison of algorithms for retrieving top-1 results for queries of size q=5 on synthetic datasets of varying dimensions d. Values of N=100,000, t=1, and U=1,000 were used for each dataset.



Query time comparison of algorithms for retrieving top-1 results for queries of varying sizes q on a 10-dimensional synthetic dataset having 100,000 points. Values of t=1 and U=1,000 were used for the dataset.



Query time comparison of algorithms for retrieving top-1 results for queries of size q=5 on 25-dimensional synthetic datasets of varying sizes N. Values of t=1 and U=1,000 were used for each dataset.



Query time analysis of ProMiSH algorithms for retrieving top-1 results for queries of varying sizes q on 25-dimensional synthetic datasets of varying sizes N. Values of t=1 and U=200 were used for each dataset.

5.1 Quality Cost

Validated the result quality of ProMiSH-E, ProMiSHA and Virtual bR*-Tree by their average approximation ratios (AAR). ProMiSH-E and Virtual bR*-Tree perform an exact search. Therefore, they always retrieve the true top-k results, and have AAR of 1. We used the results returned by them as the ground truth. Figure 7 shows AAR computed over top- 5 results retrieved by ProMiSH-A for varying query sizes on two 32-dimensional real datasets. We observe from that AAR of ProMiSH-A is always less than 1:5. This low AAR allows ProMiSH-A to return practically useful results with a very efficient time and space complexity.

The query times of the algorithms on real datasets of varying dimensions d. We used datasets of size N=50,000 and queries of size q=4. ProMiSH-A had a query time of 3 ms, ProMiSH-E had a query time of 55 ms, and Virtual bR*-Tree had a query time of 210 seconds for 32 dimensional dataset. Comparison of query times for queries of varying sizes q on a 16-dimensional real dataset of size N=70,000 is shown in figure 15. ProMiSH-A had a query time of 5 ms, ProMiSH-E had a query time of 54 ms, and Virtual bR*-Tree had a query time of 13,352 seconds for queries of size q=5. Comparison of query times on 16-dimensional real datasets of varying sizes N for queries of size q=4 is shown iProMISH-A had a query time of 3 ms, ProMiSH-E had a query time of 49 ms, and Virtual bR*-Tree had a query time of 608 seconds for a dataset of size N=70,000.

The above results show that ProMiSH significantly outperforms state-of-the-art Virtual bR*-Tree on real datasets of all dimensions and sizes and on queries of all sizes. ProMiSH-E is five orders of magnitude faster than Virtual bR*-Tree for queries of size q=5 on a 16-dimensional real dataset of size 70,000. ProMiSH-E is also at least four orders of magnitude faster than Virtual bR*-Treefor a queries of size q=4 on a 32- dimensional real dataset of size 50,000. ProMiSH-A always has an order of magnitude better performance than ProMiSHE.

The index space of ProMiSH is independent of the dimension, whereas the dataset space grows linearly with it. Therefore, the space ratio of ProMiSH decreases with dimension. The index space of Virtual bR*-Tree also grows with dimension. Therefore, ProMiSH has a lower space ratio than Virtual bR*-Tree for high dimensions

Conclusion

In this paper, we proposed solutions for the problem of top-k nearest keyword set search in multi-dimensional datasets. We developed an exact (ProMiSH-E) and an approximate (ProMiSH-A) method. We designed a novel index based on random projections and hashing. Index is used to find subset of points containing the true results. We also proposed an efficient solution to query results from a subset of data points. Our empirical results show that ProMiSH is faster than state-of-theart tree-based technique, having performance improvements of multiple orders of magnitude. These performance gains are further emphasized as dataset size and dimension increase, as well as for large query sizes. ProMiSH-A has the fastest query time. We empirically observed a linear scalability of ProMiSH with the dataset size, the dataset dimension, the query size, and the result size. We also observed that ProMiSH yield practical query times on large datasets of high dimensions for queries of large sizes.

REFERENCES

1. Z. Dou, R. Song, and J.-R. Wen, “A Large-Scale Evaluation and Analysis of Personalized Search Strategies,” Proc. Int’l Conf. World Wide Web (WWW), pp. 581-590, 2007.
2. J. Teevan, S.T. Dumais, and E. Horvitz, “Personalizing Search via Automated Analysis of Interests and Activities,” Proc. 28th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 449-456, 2005.
3. M. Spertta and S. Gach, “Personalizing Search Based on User Search Histories,” Proc. IEEE/WIC/ACM Int’l Conf. Web Intelligence (WI), 2005.
4. B. Tan, X. Shen, and C. Zhai, “Mining Long-Term Search History to Improve Search Accuracy,” Proc. ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD), 2006.
5. K. Sugiyama, K. Hatano, and M. Yoshikawa, “Adaptive Web Search Based on User Profile Constructed without any Effort from Users,” Proc. 13th Int’l Conf. World Wide Web (WWW), 2004.
6. X. Shen, B. Tan, and C. Zhai, “Implicit User Modeling for Personalized Search,” Proc. 14th ACM Int’l Conf. Information and Knowledge Management (CIKM), 2005.
7. X. Shen, B. Tan, and C. Zhai, “Context-Sensitive Information Retrieval Using Implicit Feedback,” Proc. 28th Ann. Int’l ACM SIGIR Conf. Research and Development Information Retrieval (SIGIR), 2005.
8. F. Qiu and J. Cho, “Automatic Identification of User Interest for Personalized Search,” Proc. 15th Int’l Conf. World Wide Web (WWW), pp. 727-736, 2006.
9. J. Pitkow, H. Schu¨ tze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, “Personalized Search,” Comm. ACM, vol. 45, no. 9, pp. 50-55, 2002.
10. Y. Xu, K. Wang, B. Zhang, and Z. Chen, “Privacy-Enhancing Personalized Web Search,” Proc. 16th

Int'l Conf. World Wide Web (WWW), pp. 591-600, 2007.

11. A. Krause and E. Horvitz, "A Utility-Theoretic Approach to Privacy in Online Services," J. Artificial Intelligence Research, vol. 39, pp. 633-662, 2010.
12. J.S. Breese, D. Heckerman, and C.M. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Proc. 14th Conf. Uncertainty in Artificial Intelligence (UAI), pp. 43-52, 1998.