Maximal Intersection Queries in Randomized Graph Models

Benjamin Hoffmann[1], Yury Lifshits[2⋆], and Dirk Nowotka[1]

[1] FMI, Universität Stuttgart, Germany, hoffmann,nowotka@fmi.uni-stuttgart.de
[2] Steklov Institute of Mathematics at St.Petersburg, Russia,
yura@logic.pdmi.ras.ru

Abstract. Consider a family of sets and a single set, called query set. How can one quickly find a member of the family which has a maximal intersection with the query set? Strict time constraints on the query and on a possible preprocessing of the set family make this problem challenging. Such maximal intersection queries arise in a wide range of applications, including web search, recommendation systems, and distributing on-line advertisements. In general, maximal intersection queries are computationally expensive. Therefore, one needs to add some assumptions about the input in order to get an efficient solution. We investigate two well-motivated distributions over all families of sets and propose an algorithm for each of them. We show that with very high probability an almost optimal solution is found in time logarithmic in the size of the family. In particular, we point out a threshold phenomenon on the probabilities of intersecting sets in each of our two input models which leads to the efficient algorithms mentioned above.

1 Introduction

The *nearest neighbor* problem is the task to determine in a general metric space a point that is closest to a given query point. This kind of queries appear in a huge number of applied problems: text classification, handwriting recognition, recommendation systems, distributing on-line advertisements, near-duplicate detection, and code plagiarism detection.

In this paper we consider the nearest neighbor problem in a "binary" form. Namely, every object is described as a set of its features and similarity is defined as the number of common features. For some cases, like recommending a person who has a maximal number of joint friends with you but is not your direct friend, this formalization is quite natural. On the other hand, weighted models could be simply reduced to the binary form. Let us illustrate this reduction by example. Assume that we are working with documents and their terms, and one particular term is "Ekaterinburg". Then we can introduce 8 new artificial terms Ekaterinburg1, ... Ekaterinburg8. For an object having the largest weight for

⋆ Partially supported by grants INTAS YSF 1000014-6233, INTAS 04-77-7173 and NSh-8464.2006.1

Ekaterinburg in weighted representation we simply put all eight new terms, while for objects with small weights we put only Ekaterinburg1. Thus, an intersection similarity for the new representation is somehow reflecting the classical scalar product similarity for the vector model.

In order to construct an efficient solution some assumptions should be added to the problem. In our paper we assume that the input is taken from some predefined distribution. Then we construct an algorithm and show that the time complexity and/or the accuracy are reasonably good *with high probability*. Here we use the probability over the input distribution, not over random choices of the algorithm. This probabilistic approach was inspired by the recent survey of Newman [7]. He gives a comprehensive survey about random models of graphs that agree well with many real life networks, including Web graph, friendship graphs, co-authorship graphs, and many others. Hence, we can attack the nearest neighbor problem in already "verified" random models.

The Maximal Intersection Problem. Consider a family of sets and a single set. We ask for a member of the set family which has a maximal intersection with the query set.

The Maximal Intersection Problem (MaxInt)

Database: A family $\mathcal{F}$ of n sets such that $|f| \leq k$ for all $f \in \mathcal{F}$.
Query: Given a set f_{new} with $|f_{new}| \leq k$, return $f_i \in \mathcal{F}$ with maximal $|f_{new} \cap f_i|$.
Constraints: Preprocessing time $n \cdot polylog(n) \cdot poly(k)$ or $n^{1+o(1)}$.
Query time $polylog(n) \cdot poly(k)$ or at most $o(n)$.

Let us restate the MaxInt problem in a graph theoretical notation. A database is a bipartite graph with vertex set partition (V, V') such that $|V| = n$ and the degree of every $v \in V$ is at most k. A query is a (new) vertex v (together with edges connecting v with V') of degree at most k. The query task is to return a vertex $u \in V$ with a maximal number of paths of length 2 from v to u.

Results. Inspired by [4] we present for the first time a theoretical investigation of MaxInt. In Section 2 and Section 3 we propose two new randomized models of bipartite graphs, called the Zipf model and the hierarchical schema. Assume that the terms of a query document are ordered by their frequency in the document collection. Now consider the probability curves for the two following events with parameter q. *Any q-match*: there is a document in the random (according to our models) collection that has at least q common terms with the query document. *Prefix q-match*: there is a document in the random collection that has at least q "top" terms of the query document. Both curves have the similar structure: the probability is close to 1 for small q, but suddenly, at some "magic level", it falls to nearly zero and remains so till the end. Our main observation is that these magic levels for prefix match and any match are very close to each other. And

this is extremely important for solving MaxInt. Indeed, finding the best prefix match is computationally feasible, but we don't know the general solution for MaxInt. We show that closeness of magic levels for any match and prefix match with high probability allows to find an approximate solution for MaxInt.

Related Work. MaxInt is a special case of the nearest neighbor problem. Indeed, one just needs to define the distance between two vertices in a bipartite graph as the inverse of the number of 2-step paths between these vertices. Yianilos proposed in [9] *vp*-trees (vantage point trees), a data structure that solves nearest neighbor problem on general metric spaces. The preprocessing time of his algorithms is in $\mathcal{O}(n \cdot \log n)$ and expected query time is in $\mathcal{O}(\log n)$. However, he uses a strong anti-discrete assumption about the metric that does not hold for our similarity-by-intersection-size model.

Nearest neighbors are particularly well studied in vector models with a similarity function based on the scalar product [3]. Actually, we can interpret a document as a vector of 0s and 1s (1 means a term is contained in a document). Then, the scalar product is equal to the size of the intersection. Unfortunately, the first algorithm from [3] needs quadratic space and the second one has linear query time, so none of them preserve our constraints. Both algorithms return a γ-approximate nearest neighbor and are based on random projections of the database vectors onto several randomly chosen directions.

Closely related to MaxInt is *text search*. Finding documents that most fit to some given search terms can also be considered as a problem on a bipartite graph. The documents and terms are the nodes and edges are drawn when a term occurs in a document. Basically the task is to find all documents containing *every* query term and rank these documents by relevance. The key technique in this area is inverted files (inverted indexing). A comprehensive survey of the topic can be found in [10].

A problem similar to MaxInt called *collaborative filtering* is considered by O'Connor and Herlocker in [8]. Collaborative filtering can be seen as a bipartite graph problem where nodes represent people and objects and weighted edges between these people and objects are defined by ratings. The task is to estimate the weight of a new edge.

2 MaxInt in the Zipf Model

Throughout the following sections we use a documents-terms notation, that is, vertices from $\mathcal{D} = \{d_1, \ldots, d_n\}$ represent documents and vertices from $\mathcal{T} = \{t_1, \ldots, t_m\}$ represent terms. Let $n \geq 3$, and $m \leq \mathit{poly}(n)$. By log we always mean $\log_2$, while ln denotes $\log_e$.

We now describe a probabilistic mechanism for generating a document collection called the *Zipf model*. Every document is generated independently. Term occurrences are also independent. A document contains term t_i with probability $\frac{1}{i}$. Hence, the expected number of terms in a document is approximately equal to $\ln m$ in our model. This model is similar to the *configuration model* (see [7])

with Zipf's law for distribution of term degrees and constant document degrees. Zipf's law states that in natural language texts the frequency of a word is approximately inversely proportional to its rank in the frequency table[1]. For more details about Zipf's law see [6].

Remark 1. The frequency of a term t in a collection $\mathcal{D}$ of documents is defined as

$$\frac{|\{d \in \mathcal{D} \mid t \in d\}|}{|\mathcal{D}|}.$$

The expected frequency of the term t_i is equal to $\frac{1}{i}$. At the same time, the expected frequency rank for t_i is exactly the i-th value among those of all terms. So the Zipf model reflects in a natural way Zipf's law. Since some of our motivating applications also deal with natural language texts, we can state that the Zipf model agrees with real life at least by degree distribution.

Remark 2. In the following proofs we will use two inequalities ($a, b > 0$):

$$\left(1 - \frac{a}{b}\right)^b < 2^{-a}, \qquad a \leq b \qquad (*)$$

$$\left(1 - \frac{1}{ab}\right)^a > 1 - \frac{1}{b}, \qquad a, b \geq 2 \qquad (**).$$

These inequalities follow from the well-known fact $\lim\limits_{n\to\infty}(1 - \frac{x}{n})^n = e^{-x}$.

For further considerations we partition the set of terms in the following way:

$$\underbrace{t_1\ t_2}_{P_1} \quad \underbrace{t_3\ t_4\ t_5\ t_6\ t_7}_{P_2} \quad \dots$$

Here the group P_i includes terms from $t_{\lceil e^{i-1}\rceil}$ to $t_{\lfloor e^i \rfloor}$ A document that contains $\ln m$ terms $p_1 \dots p_{\ln m}$, $p_i \in P_i$, will be called *regular*.

We now introduce a magic level to give statements about the most probable size of maximal intersection

$$q = \sqrt{2 \ln n}.$$

Theorem 1 (Magic Level for the Zipf Model). *Let $3 \leq \gamma < q - 1$ be a positive integer. Fix n, m and a regular query document d_{new}. Then for a document collection following the Zipf model the following holds:*

1. *The probability that there exists a document $d \in \mathcal{D}$ that contains the first $q - \gamma$ terms ("prefix match") of d_{new} is greater than $1 - 2^{-e^{\frac{q(\gamma+1)}{2}}}$.*
2. *The probability that there exists a document $d \in \mathcal{D}$ that contains at least $q + \gamma$ terms ("any match") of d_{new} is smaller than $\frac{1}{e^{(\gamma-2)q-1}}$.*

[1] In the *frequency table*, the most frequent term is at rank 1, the second most frequent term at rank 2 and so on.

Proof. 1. The probability that a document contains the prefix of length $q-\gamma$ of d_{new} is equal to or greater than

$$\frac{1}{e}\cdot\ldots\cdot\frac{1}{e^{q-\gamma}} > \frac{1}{e^{\frac{(q-\gamma+1)^2}{2}}} > \frac{e^{\frac{q(\gamma+1)}{2}}}{e^{\frac{q^2}{2}}} = \frac{e^{\frac{q(\gamma+1)}{2}}}{n}.$$

This means the probability that there exists no document in $\mathcal{D}$ that contains the $(q-\gamma)$-prefix of d_{new} is equal to or smaller than

$$\left(1-\frac{e^{\frac{q(\gamma+1)}{2}}}{n}\right)^n < 2^{-e^{\frac{q(\gamma+1)}{2}}},$$

which follows from inequality $(*)$ (note that $e^{\frac{q(\gamma+1)}{2}} < n$ for $\gamma < q-1$). So with probability greater than $1-2^{-e^{\frac{q(\gamma+1)}{2}}}$ there exists a document $d \in \mathcal{D}$ that has all terms from the $(q-\gamma)$-prefix of d_{new}.

2. Let $d_{new} = \{a_1, \ldots, a_{\lceil \ln m \rceil}\}$, $s = q+\gamma$. We now estimate the probability that a random document has a large overlap with d_{new}:

$$\begin{aligned}
Pr(|d_{new}\cap d| \geq s) &\leq \sum_{j_1<\cdots<j_s} Pr(a_{j_1},\ldots,a_{j_s}\in d)\\
&\leq \sum_{j_1<\cdots<j_s} \exp(-\sum_{k=1}^{s}(j_k-1))\\
&= \sum_{\Delta_1\geq 0,\,\Delta_2,\ldots\Delta_s>0} \exp(-s\Delta_1-(s-1)\Delta_2-\cdots-\Delta_s)\\
&\leq \sum_{\Delta_1=0}^{\infty}\exp(-s\Delta_1)\cdot\prod_{k=2}^{s}\sum_{\Delta_k=1}^{\infty}\exp(-\Delta_k(s-k+1))\\
&= \frac{1}{1-\exp(-s)}\prod_{k=2}^{s}\frac{\exp(-(s-k+1))}{1-\exp(-(s-k+1))}\\
\leq \exp(1-\sum_{k=2}^{s}s-k) &\leq \exp\left(1-\frac{(s-2)^2}{2}\right) \leq \frac{1}{e^{\frac{q^2}{2}}\cdot e^{(\gamma-2)q-1}} \leq \frac{1}{n\cdot e^{(\gamma-2)q-1}}.
\end{aligned}$$

The probability that not a single document in $\mathcal{D}$ contains at least $q+\gamma$ terms of d_{new} is equal to or greater than

$$\left(1-\frac{1}{n\cdot e^{(\gamma-2)q-1}}\right)^n > 1-\frac{1}{e^{(\gamma-2)q-1}},$$

which follows from inequality $(**)$ (note that $e^{(\gamma-2)q-1}\geq 2$ for $3\leq\gamma<q$). Therefore, we proved that the probability that any document matches d_{new} at $q+\gamma$ arbitrary positions is smaller than $\frac{1}{e^{(\gamma-2)q-1}}$.

By Theorem 1 we can conclude that with very high probability there exists a document in $\mathcal{D}$ that matches a prefix of length $q-\gamma$, whereas with quite small probability there exists a document that has at least $q+\gamma$ common terms with d_{new} (at arbitrary positions). We get the following algorithm:

MaxInt Algorithm in the Zipf Model

Preprocessing:

1. For every document: Sort the term list according to the position of the term in the frequency table.
2. For every document: Generate the set of all possible *regular* $(q-\gamma)$-*lists*, i.e. generate all possible term subsets of size $(q-\gamma)$ containing one term from every group $P_1, \ldots, P_{q-\gamma}$.
3. Sort these regular lists and store for every list a pointer to the corresponding document.

Query: Find a regular $(q-\gamma)$-list having the maximal common prefix with the query document by binary search. Return the document corresponding to this $(q-\gamma)$-list.

The running time is as follows (for *average* case[2] analysis we assume that the length of term lists is $\log m$, for *worst* case analysis the length is m):

	average	worst
Step 1	$\mathcal{O}(n \cdot \log m \cdot \log\log m)$	$\mathcal{O}(n \cdot m \cdot \log m)$
Step 2	$n^{1+o(1)}$	$\mathcal{O}(n^2 \cdot \log n)$
Step 3	$\mathcal{O}(\log m \cdot n \cdot \log n)$	$\mathcal{O}(m \cdot n \cdot \log n)$
Query	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(\log^2 n)$

Let us explain the estimations from the second line. The number of all possible regular $(q-\gamma)$-lists is equal to

$$|P_1| \cdot \cdots \cdot |P_{q-\gamma}| \leq \prod_{k=1}^{q-\gamma} e^k < e^{q^2/2} = n$$

Therefore, a single document can generate at most n different regular $(q-\gamma)$-lists, the $\log n$ factor arises from the size of a single list. Let us prove the bound for the average case. The probability of containing some fixed regular $(q-\gamma)$-list is $n^{-1+o(1)}$. Summing over all possible lists we see that the expected number of generated regular lists per document is at most $n^{-1+o(1)} \cdot n = n^{o(1)}$. Therefore, the expected time for the indexing stage is $n^{1+o(1)}$.

[2] Only for the average case our constraints from Section 1 are preserved.

One can try to improve the accuracy of our algorithm by finding a "maximal prefix with at most one difference to the query document". A famous technique called "indexing with errors" [2, 5] might be useful for such an extension.

3 MaxInt in the Hierarchical Schema

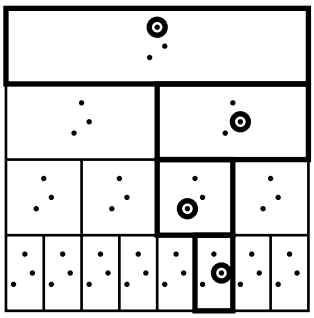

Fig. 1. Hierarchical Schema

Let $|\mathcal{D}| = 2^k$, $k \in \mathbb{N}$, $k \geq 3$, and $|d| = k$ for every $d \in \mathcal{D}$. Let $|\mathcal{T}| = (2^k - 1) \cdot k$ be the number of different terms. A *hierarchical schema* is a table with k levels, level 1 to level k. Level i is divided to 2^{i-1} cells, $1 \leq i \leq k$. Every cell contains k terms. A document collection based on this schema can be generated as follows: Choose a random cell on level k. Then mark all cells that are above it and choose one random term in every marked cell. Now each document corresponds to a path from the top cell to a bottom cell (see Figure 1).

Remark 3. We claim that the hierarchical schema follows *Zipf's law*. To be more precise, the following holds: For every level the product of expected frequency and expected frequency rank of a term is the same. Indeed, the expected frequency of a term on level i is calculated by the formula $\frac{2^k}{2^{i-1} \cdot k}$. The expected rank of a term is calculated by the formula $(2^{i-1} - 1) \cdot k + 2^{i-2} \cdot k$. Hence, the product between frequency and frequency rank (divided by 2^k) is equal to

$$\frac{2^k}{2^{i-1} \cdot k} \cdot (\frac{1.5}{2^k} \cdot 2^{i-1} - 1) \cdot k \approx \frac{3}{2}$$

and hence Zipf's law applies.

Again we introduce *magic levels* to give statements about the most probable size of maximal intersection. The magic levels are

$$q = \frac{k}{1 + \log k} \qquad \text{and} \qquad q' = \frac{k}{\log k}.$$

Theorem 2 (Magic Levels for Hierarchical Schema). *Let $k \geq 4$ and $2 \leq \gamma < q$ be a positive integer. Fix a query document d_{new} following hierarchical schema and take a randomly generated document collection $\mathcal{D}$:*

1. *The probability that there exists a document $d \in \mathcal{D}$ that matches the same terms from top $q-\gamma$ levels ("prefix match") of d_{new} is greater than $1-2^{-(2k)^\gamma}$.*
2. *The probability that there exists a document $d \in \mathcal{D}$ that matches at least $q' + \gamma$ terms ("any match") of d_{new} is smaller than $\frac{2}{k^{\gamma-1}}$.*

Proof. 1. The number of different prefixes of length $q-\gamma$ is equal to

$$k(2k)^{q-\gamma-1} \;<\; 2^{(1+\log k)(q-\gamma)} \;=\; 2^{(1+\log k)(\frac{k}{1+\log k}-\gamma)} \;=\; 2^k \cdot (2k)^{-\gamma}.$$

So the probability that a new document does not match a prefix of length $q-\gamma$ with any document from $\mathcal{D}$ is smaller than

$$\left(1-\frac{(2k)^\gamma}{2^k}\right)^{2^k} \;<\; 2^{-(2k)^\gamma}.$$

Since $(2k)^\gamma < 2^k$, this inequality follows from inequality $(*)$. We get that the probability that there exists a document with the same prefix as d_{new} of length $q-\gamma$ is greater than $1-2^{-(2k)^\gamma}$.

2. Let $t \geq q'+\gamma$ be the last position where the terms of d and d_{new} match. We want to estimate the probability that d_{new} matches at least $q'+\gamma$ terms at arbitrary positions with d. The probability that the first t terms (beginning at the top) of d and d_{new} are all in the same cells is 2^{1-t}. The probability that at least $q'+\gamma$ terms are matched on some fixed positions is equal or smaller than $\left(\frac{1}{k}\right)^{q'+\gamma} \cdot \left(\frac{k-1}{k}\right)^{t-q'-\gamma}$. An upper bound for the number of different possibilities of matching at least $q'+\gamma$ out of t terms is 2^t. Since the factor $\left(\frac{k-1}{k}\right)^{t-q'-\gamma}$ is smaller than 1, overall we get that the probability that d_{new} matches at least $q'+\gamma$ terms at arbitrary positions with d is equal to or smaller than

$$k \cdot 2^t \cdot \left(\frac{1}{k}\right)^{q'+\gamma} \cdot 2^{1-t} \;=\; 2 \cdot k \cdot \left(\frac{1}{k}\right)^{q'+\gamma} \;=\; 2 \cdot \left(\frac{1}{k}\right)^{q'+\gamma-1}.$$

The factor k is due to the fact that we need to consider all possible levels for the last matched position t. Now the probability that no document matches at $q'+\gamma$ arbitrary positions with d_{new} is greater than

$$\left(1-2\cdot\left(\frac{1}{k}\right)^{q'+\gamma-1}\right)^{2^k} \quad = \quad \left(1-\frac{1}{2^k \cdot \frac{k^{\gamma-1}}{2}}\right)^{2^k} \quad > \quad 1 \;-\; \frac{2}{k^{\gamma-1}},$$

which follows from inequality $(**)$, since $\gamma \geq 2, k \geq 4$. So the probability that we get any match in $\mathcal{D}$ is smaller than $\frac{2}{k^{\gamma-1}}$.

By Theorem 2 we get an analogue algorithm as the one for the Zipf model:

MaxInt Algorithm in the Hierarchical Schema

Preprocessing:

1. For every document: Sort the term list according to the hierarchical schema, i.e. according to the levels in which the terms appear.
2. Sort the documents according to their corresponding cell paths, i.e. documents that correspond to the leftmost path in the schema are at the beginning of the sorted list. Documents that correspond to the same cell path are sorted lexicographically.

Query: Find a document having the maximal common prefix with the query document by binary search.

Step 1 of preprocessing needs time $\mathcal{O}(2^k \cdot k \cdot \log k)$. Step 2 needs time $\mathcal{O}(2^k \cdot k^2)$. So the overall running time of the preprocessing is in $\mathcal{O}(2^k \cdot k^2)$. The query time is in $\mathcal{O}(k^2)$.

4 Further Work

In this paper we have shown that assumptions on the random nature of the input can lead to *provable* time and accuracy bounds for MaxInt. Also, we have discovered a MaxInt threshold phenomenon in two randomized models.

The next step is to understand it better. Does it hold for other randomized models from [7], especially the preferential attachment model? Is it still true when we replace the Zipf distribution by a power law distribution? Does it hold in the real life networks? Can we introduce randomized models for sparse vector collections and find a similar effect there? Of course, the most challenging problem is to find an exact algorithm for MaxInt (preserving our time constraints) or to prove its hardness. What are other particular cases or assumptions that have efficient MaxInt solutions? On the other hand, we have a very particular subcase for which we still do not believe in a positive solution. Hence, we ask for a hardness proof for the following *on-line inclusion problem*. Note that we have a constraint on space for preprocessing, not time. A related problem but with a much stronger restriction on preprocessing space was proven to be hard by Bruck and Naor [1].

On-line Inclusion Problem

Database: A family $\mathcal{F}$ of 2^k subsets of $[1 \dots k^2]$.

Query: Given a set $f_{new} \subseteq [1 \dots k^2]$, decide whether there exists an $f \in \mathcal{F}$ such that $f_{new} \subseteq f$.

Constraints: Space for preprocessed data $2^k \cdot poly(k)$.
Query time $poly(k)$.

Our algorithm in Section 3 uses polylogarithmic time (in the number of documents) but it returns only an approximate solution with high probability (not every time). Can we get an optimal solution or at least a *guaranteed* approximation by relaxing the time constraint to *expected* polylogarithmic time?

Acknowledgments. The authors would like to thank Holger Petersen, Hinrich Schütze, Kirill Shmakov and Jason Utt for their useful comments and fruitful discussions.

References

1. J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, 1990.
2. R. Cole, L.-A. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *STOC '04*, pages 91–100, 2004.
3. J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC '97*, pages 599–608, 1997.
4. Y. Lifshits. A Guide to Web Research. Materials of mini-course at Stuttgart University. Available at `http://logic.pdmi.ras.ru/~yura/webguide.html`, 2007.
5. M. G. Maaß and J. Nowak. A new method for approximate indexing and dictionary lookup with one error. *Inf. Process. Lett.*, 96(5):185–191, 2005.
6. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2002.
7. M. Newman. The structure and function of complex networks. SIAM Review, 45(2):167– 256, 2003.
8. M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. SIGIR '01, Workshop on Recommender Systems., 2001.
9. P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA '93*, pages 311–321, 1993.
10. J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6 (Article No.), 2006.