Approximating All-Pairs Similarity Search by Rademacher Average

Shiyu Ji
shiyu@cs.ucsb.edu

1 Introduction

All-pairs similarity search (APSS) has received extensive research interest recently [5, 35, 1, 30]. To improve performance, many approximation approaches have been proposed [13, 11, 17, 10]. This paper considers two approximation methods for cosine similarity based search [29, 35, 1, 30], and SimRank based search [18, 20, 12, 19] respectively. The approximation error of each our algorithm is upper bounded by using Rademacher average [4, 23, 3].

2 Related Works

All-pairs similarity search (APSS) is recently a popular research topic in the community of information retrieval [5, 35, 1, 30]. Given a big set of objects, the goal of APSS is to efficiently compute (or approximate) the similarities between each pair of objects. Many similarity measures have been proposed [29], e.g., Cosine Similarity [31], SimRank [18], Jaccard Index [14], Metric Distance [29], Pearson Correlation [6], etc. For similarity search, Cosine Similarity and SimRank are very popular ([31, 35, 1, 30] for Cosine, and [20, 12, 19, 36] for SimRank). A major challenge of APSS is the large volume of computation: given n objects, without any assumption on the similarity distribution (e.g., the similarity matrix is sparse), the time complexity is at least $O(n^2)$. To compute more efficiently, the state-of-art research works often use parallelism [9, 15] and dissimilarity detection [1, 30]. If only approximations are needed, how to efficiently sample with negligible error for similarity search is another research focus [13, 11, 17, 10]. This paper focuses on the approximation problem.

We may treat the similarity approximation as a learning problem over large scaled data, and we need to upper bound the learning error for the worst case. In statistical learning theory, such upper bounds are usually called risk bounds [32]. There are two classical methods to compute the risk bounds: by VapnikChervonenkis (VC) dimension [33, 34, 32] and by Rademacher average [23, 4, 3]. Many approximation algorithms that use these two techniques have been proposed [25, 26, 27, 28]. Riondato and Kornaropoulos [25, 26] proposed algorithms that use VC dimension to upper bound the sample size that is sufficient to approximate the betweenness centralities [8] of all nodes in a graph with guaranteed error bounds. One limitation of the VC-based algorithms is that the upper bounds of some characteristic quantities (e.g., the maximum length of any shortest path) are needed [25, 26, 28]. But such bounds are not always available. One year later, Riondato and Upfal used Rademacher average to approximate the frequent itemsets [27] and betweenness centralities [28]. They also found that by using Rademacher average, we can avoid the aforementioned limitation of VC-based solutions. It has been proved that Rademacher average can be applied to various approximation problems, which are out of the scope of classical learning framework. This paper basically follows this idea. To the best of our knowledge, there is no research work connecting similarity approximation with Rademacher average. We attempt to fill this void.

3 Problem Formulation and Preliminaries

3.1 Cosine Similarity

We consider the cosine similarity based all-pairs similarity search. Suppose there are n vectors (each vector can represent a user profile or a web page). Each vector contains m non-negative features. Define the cosine similarity between two vectors u and v as

$$Sim(u, v) = \frac{1}{||u|| \cdot ||v||} \sum_{i=1}^{m} u_i \cdot v_i,$$

where u_i, v_i denotes the i-th feature value of u, v. For simplicity, we assume all the vectors are adjusted with the same norm: $||v|| = \sqrt{m}$ for every v in the n vectors. Then the equation above can be simplified as

$$Sim(u, v) = \frac{1}{m} \sum_{i=1}^{m} u_i \cdot v_i.$$

That is, the similarity is defined as the average on the corresponding feature products between the vectors.

To do the all-pairs similarity search (APSS), we need to compute the similarity between each pair of vectors. Since there are $n(n-1)$ pairs, and for each pair we need m times of multiplication, the total complexity of a naiive algorithm is $O(n^2 m)$. Fortunately, there are many methods to detect dissimilar pairs (two vectors without sharing any feature) [1, 30, 21], which can save a lot of computation. For the state-of-art works, to compute all the pairs, the complexity can be lowered to $O(nkm)$, where k is much less than n. However, to the best of our knowledge, there were few discussions on the size of features m. If we can only consider a part of the m features without significantly deteriorating the accuracy, then the total computation time for APSS can be lowered significantly (note that nk is still large for very big dataset). We can approximating the similarity by sampling on the features.

3.2 SimRank

SimRank [18] is a popular measure of the similarity between two nodes in a graph based on the idea that similar nodes are often referred by other similar nodes. Denote by $s(a, b)$ the SimRank between nodes a and b. If $a = b$, then $s(a, b)$ is defined to be 1. Otherwise,

$$s(a, b) = \frac{c}{|I(a)| \cdot |I(b)|} \sum_{i \in I(a)} \sum_{j \in I(b)} s(i, j),$$

where c is a constant in $(0, 1)$ and $I(a)$ denotes the in-neighbors of a. If there is an edge from a to b, then a is an in-neighbor of b. An important fact is that SimRank can be equivalently built upon Random Surfer-Pairs Model [18]. That is, $s(a, b)$ can also be written as follows:

$$s(a, b) = \sum_{t:(a,b)(x,x)} P[t] \cdot c^{L(t)},$$

where t is a pair of random walks with the same number of steps, which lead a and b to meet at one node x, and $P[t]$ denotes the probability when t is chosen, and $L(t)$ denotes the number of steps of each walk in t. Here we take random walks from the reversed graph, in which each edge is reversed compared with the original graph. A random walk works as follows: in the tour at any node, which has k out-edges, we take one of the k edge with equal probability $1/k$ as our next step. Note that as the length of random walk grows, its contribution to $s(a, b)$ decreases exponentially. Hence in practice, we only need to consider relatively short walks, e.g., with length no more than T.

3.3 Rademacher Average

This section proposes our approximation algorithm and its analysis. Suppose we want to compute the cosine similarities between a fixed vector u and other vectors v_1, v_2, $\cdots$, v_n. We take the m features as the sample space S:

$$S = \{s_1, \cdots, s_k\} \subseteq D$$

where k is the number of samples we take, and D is the feature space: $D = \{1, 2, \cdots, m\}$. Let F be a collection of functions from the features D to the interval $[0, m]$. For each function $f \in F$, define the *true average* $A_D(f)$ and *sampled average* $A_S(f)$ as follows:

$$A_D(f) = \frac{1}{m}\sum_{i=1}^{m} f(i), \quad A_S(f) = \frac{1}{k}\sum_{i=1}^{k} f(s_i).$$

Define the *uniform deviation* [24] of F given S as

$$U_S(F) = \sup_{f \in F}[A_S(f) - A_D(f)].$$

Note that if F is a finite set (in this paper we will see this is true), supreme can be replaced by maximum:

$$U_S(F) = \max_{f \in F}[A_S(f) - A_D(f)].$$

Define the *Rademacher average* [23, 4, 24] of F given S as

$$R_S(F) = \mathbb{E}_\sigma \left[\sup_{f \in F} \frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) \right],$$

where each σ_i is a random variable uniformly distributed over $\{-1, 1\}$, and the mean $\mathbb{E}_\sigma$ takes randomness over all the σ_i's conditionally on S. Again, one can replace supreme by maximum.

The main motivation of our algorithm is that the difference between true average and sampled average is upper bounded by the Rademacher average as follows. We leave the proofs to the Appendix.

Theorem 1. Let F be a collection of functions f mapping D to $[0, m]$. With probability at least $1 - \delta$, we have

$$\sup_{f \in F} |A_S(f) - A_D(f)| \leq R_S(F) + \left(m + m\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + m\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{8R_S(F)}{m}} \right) \sqrt{\frac{\log\frac{8}{\delta}}{2k}}.$$

The remaining item we need to upper bound is $R_S(F)$. Our result is similar to Massart's lemma [2].

Theorem 2.

$$R_S(F) \leq \frac{\ell}{k}\sqrt{8\log|F|},$$

where $\ell^2 = \sup_{f \in F} \sum_{i=1}^{k} f(s_i)^2$.

By the above two theorems, we can bound our approximation error even for the worst case.

4 Approximating Cosine Similarities

The approximation algorithm takes as input a collection of n vectors V and two parameters (ϵ, δ) whose values are between 0 and 1. The algorithm outputs a set $C = \{\tilde{S}(u, v) : u, v \in V, u \neq v\}$, where $\tilde{S}(u, v)$ is the (ϵ, δ)- approximation of cosine similarity between u and v, i.e., with probability at least $1 - \delta$, the worst approximation error in C is at most ϵ. Each vector in V contains m features. We take the m features as

Algorithm 1 Cosine Similarity Approximation

Input: vectors V ($|V| = n$); features U ($|U| = m$); $\epsilon \in (0,1)$; $\delta \in (0,1)$.
Output: $Sim(u, v)$ for each $u, v \in V$ s.t. $u \neq v$.

$k \leftarrow 0$;
$S \leftarrow \emptyset$;
$S(u, v) \leftarrow 0$ for each $u, v \in V$ s.t. $u \neq v$;
while $k \leq m$ **do**
  $k' \leftarrow \mathsf{next-sample-size}(k)$;
  for i from k to $k' - 1$ **do**
    Uniformly sample a feature s from $U \setminus S$;
    $S(u, v) \leftarrow S(u, v) + u[s] \cdot v[s]$;
    $S \leftarrow S \cup \{s\}$;
  end for
  $\Delta \leftarrow \frac{4\ell\sqrt{\log n}}{k} + \left(m + m\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + m\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{32\ell\sqrt{\log n}}{mk}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}}$;
  if $\Delta \leq \epsilon$ **then**
    break the while loop;
  end if
end while
$\mathsf{S} \leftarrow \{S(u, v)/k : u, v \in V, u \neq v\}$.
return S.

the sample space $D = \{1, \cdots, m\}$. For each feature $s \in D$, let $f_{u,v}(s) = u_s \cdot v_s$, where u_s is the s-th feature value of the vector u. Let $F = \{f_{u,v} : u, v \in V, u \neq v\}$. Thus $|F| < n^2/2$ since symmetry of cosine similarity. It is clear that the true average of $f_{u,v}$ equals to the cosine similarity between u and v. Given the sampled features, the upper bound of $R_S(F)$ is

$$R_S(F) \leq \frac{\ell}{k}\sqrt{8\log|F|} < \frac{4\ell\sqrt{\log n}}{k},$$

where $\ell = \sqrt{\max_{f\in F}\sum_{i=1}^{k} f(s_i)^2}$. Since F is finite, we can replace supreme by maximum.

The approximation algorithm works in an iterative mode. For each iteration, we sample some new features s_i's among D and aggregate the feature products given by $f_{u,v}(s_i)$ for each $f_{u,v}$ in F. Then we compute the error upper bound:

$$\Delta = \frac{4\ell\sqrt{\log n}}{k} + \left(m + m\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + m\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{32\ell\sqrt{\log n}}{mk}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}},$$

where $\ell = \sqrt{\max_{f\in F}\sum_{i=1}^{k} f(s_i)^2}$, and k is the number of aggregated samples. If $\Delta \leq \epsilon$, then we stop and return the averages $\frac{1}{k}\sum_{s_i\in S} f_{u,v}(s_i)$ for each pair u, v. Otherwise, we continue to the next round, where we will sample more features. If $\Delta \leq \epsilon$ can never be satisfied, at the end we will sample all the m features and return the averages as the exact solution. Algorithm 1 gives the pseudocode of our solution.

It is clear to verify the correctness of our algorithm by Theorem 1 and Theorem 2.

5 Approximating SimRank

We approximate SimRank scores on the nodes in the digraph $G = (V, E)$. The basic idea is similar to the case of cosine similarity approximation, but the sample space D is changed to the $2T$-dimensional manifold $[0, 1]^{2T}$, where $[0, 1]$ is the interval of real numbers between 0 and 1, and F is now defined as

$$F = \{f_{a,b} : a, b \in V, a \neq b\},$$

Algorithm 2 Random Walk Generation

Input: reversed graph $\overline{G}$, sequence $(s_1, \cdots, s_T) \in [0,1]^T$, starting node a.
Output: a random walk $W = (a, \cdots)$ starting from a.

```
Let W be a sequence of length T + 1;
W[0] ← a;
for i from 1 to T do
    I ← {b : there is an edge in G̅ from W[i − 1] to b};
    k ← |I|;
    if k=0 then
        W[i] ← W[i − 1];
    else
        Sort I in lexicographical order: (I[0], ··· , I[k − 1]);
        x ← ⌊s_{i−1} * k⌋;
        W[i] ← I[x];
    end if
end for
return W.
```

where given any two nodes a, b, the function $f_{a,b}$ takes a sample $s_i \in D$ as input and outputs a value in the interval $[0, c]$. Algorithm 2 shows how to generate a random walk given a sample from $[0,1]^T$ and a starting node a. It is clear that the random walk chosen by Algorithm 2 is uniformly distributed if the sample is uniformly drawn from $[0,1]^T$. Thus given two nodes a, b and a sample $s_i \in [0,1]^{2T}$, we can generate two random walks W_a, W_b starting from a and b respectively. Let

$$l_{a,b}(s_i) = \min\{j : 1 \leq j \leq T, W_a[j] = W_b[j]\},$$

i.e., the number of steps before two surfers starting from a, b meet at some node. If W_a and W_b never meet within T steps, $l_{a,b}(s_i)$ is defined to be 0.

The function $f_{a,b}(s_i)$ is defined as

$$f_{a,b}(s_i) = \begin{cases} 0 & \text{if } l_{a,b}(s_i) = 0, \\ c^{l_{a,b}(s_i)} & \text{otherwise.} \end{cases}$$

It is clear that the true average $A_D(f_{a,b})$ equals to the SimRank between the nodes a and b. Thus a similar approximation idea based on Rademacher average can be applied. We still need to upper bound the worst approximation error. The new bound is stated as follows.

Theorem 3. Let F be a collection of functions f mapping D to $[0, c]$. With probability at least $1 - \delta$, we have

$$\sup_{f \in F} |A_S(f) - A_D(f)| \leq R_S(F) + \left(c + c\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + c\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{8R_S(F)}{c}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}}.$$

The proof is essentially the same as the proof of Theorem 1. We leave the discussions to the Appendix. Also the upper bound of $R_S(F)$ given by Theorem 2 remains unchanged.

Our approximation algorithm works as follows. For each iteration, we sample some new features s_i's among D, generate random walks W_a, W_b from nodes a, b given s_i and aggregate $f_{a,b}(s_i)$ for each $f_{a,b}$ in F. Then we compute the error upper bound:

$$\Delta = \frac{4\ell\sqrt{\log n}}{k} + \left(c + c\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + c\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{32\ell\sqrt{\log n}}{ck}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}},$$

Algorithm 3 SimRank Approximation

Input: reversed graph $\overline{G}$; $\epsilon \in (0,1)$; $\delta \in (0,1)$; maximum random walk length T; maximum number of iterations R; SimRank constant c.

Output: $\mathsf{SimRank}(u,v)$ for each $u,v \in V$ s.t. $u \neq v$.

$k \leftarrow 0$;
$S(u,v) \leftarrow 0$ for each $u,v \in V$ s.t. $u \neq v$;
for count from 1 to R **do**
  $k' \leftarrow \mathsf{next-sample-size}(k)$;
  for i from k to $k'-1$ **do**
    Uniformly sample a feature s from $[0,1]^{2T}$;
    for each $a,b \in V \times V$ s.t. $a \neq b$ **do**
      $W_a \leftarrow \mathsf{Random\ Walk\ Generation}(s[1\cdots T])$;
      $W_b \leftarrow \mathsf{Random\ Walk\ Generation}(s[(T+1)\cdots 2T])$;
      $L \leftarrow \{j : 1 \leq j \leq T, W_a[j] = W_b[j]\}$;
      if $|L| > 0$ **then**
        $l_{a,b} \leftarrow \min L$;
        $S(u,v) \leftarrow S(u,v) + c^{l_{a,b}}$;
      end if
    end for
  end for
  $\Delta \leftarrow \frac{4\ell\sqrt{\log n}}{k} + \left(c + c\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + c\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{32\ell\sqrt{\log n}}{ck}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}}$;
  if $\Delta \leq \epsilon$ **then**
    break the while loop;
  end if
end for
$\mathsf{S} \leftarrow \{S(u,v)/k : u,v \in V, u \neq v\}$.
if count $= R$ **then**
  show a message that (ϵ,δ) cannot be satisfied.
end if
return S.

where $\ell = \sqrt{\max_{f\in F}\sum_{i=1}^{k} f(s_i)^2}$, and k is the aggregated samples. If $\Delta \leq \epsilon$, then we stop and return the averages $\frac{1}{k}\sum_{s_i\in S} f_{a,b}(s_i)$ for each pair u,v. Otherwise, we continue to the next round, where we will sample more features. If $\Delta \leq \epsilon$ can never be satisfied with a certain number of iterations, we return the approximated SimRank scores of node pairs, with the message that the input parameters (ϵ,δ) cannot be satisfied by our algorithm.

It is clear to verify the correctness of our algorithm by Theorem 3 and Theorem 2.

References

[1] Maha Alabduljalil, Xun Tang, and Tao Yang. Cache-conscious performance optimization for similarity search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 713–722. ACM, 2013.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, and Sandro Ridella. A deep connection between the vapnik–chervonenkis entropy and the rademacher complexity. *IEEE transactions on neural networks and learning systems*, 25(12):2202–2211, 2014.

[3] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *Annals of Statistics*, pages 1497–1537, 2005.

[4] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[5] Roberto J Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, pages 131–140. ACM, 2007.

[6] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[7] S Boucheron, G Lugosi, and Pascal Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 1999.

[8] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[9] Liangliang Cao, Brian Cho, Hyun Duk Kim, Zhen Li, Min-Hsuan Tsai, and Indranil Gupta. Delta-simrank computing on mapreduce. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 28–35. ACM, 2012.

[10] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[11] Ronald Fagin, Ravi Kumar, and Dandapani Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 301–312. ACM, 2003.

[12] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, and Makoto Onizuka. Efficient search algorithm for simrank. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 589–600. IEEE, 2013.

[13] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. *VLDB*, 99(6):518–529, 1999.

[14] Lieve Hamers, Yves Hemeryck, Guido Herweyers, Marc Janssen, Hans Keters, Ronald Rousseau, and André Vanhoutte. Similarity measures in scientometric research: the jaccard index versus salton's cosine formula. *Information Processing & Management*, 25(3):315–318, 1989.

[15] Guoming He, Haijun Feng, Cuiping Li, and Hong Chen. Parallel simrank computation on large graphs with iterative aggregation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 543–552. ACM, 2010.

[16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[17] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[18] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[19] Mitsuru Kusumoto, Takanori Maehara, and Ken-ichi Kawarabayashi. Scalable similarity search for simrank. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 325–336. ACM, 2014.

[20] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 465–476. ACM, 2010.

[21] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 155–162. ACM, 2009.

[22] Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[23] Mehryar Mohri and Afshin Rostamizadeh. Rademacher complexity bounds for non-iid processes. In *Advances in Neural Information Processing Systems*, pages 1097–1104, 2009.

[24] Luca Oneto, Alessandro Ghio, Davide Anguita, and Sandro Ridella. An improved analysis of the rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44:107–111, 2013.

[25] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 413–422. ACM, 2014.

[26] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

[27] Matteo Riondato and Eli Upfal. Mining frequent itemsets through progressive sampling with rademacher averages. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2015.

[28] Matteo Riondato and Eli Upfal. Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *arXiv preprint arXiv:1602.05866*, 2016.

[29] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

[30] Xun Tang, Maha Alabduljalil, Xin Jin, and Tao Yang. Load balancing for partition-based similarity search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 193–202. ACM, 2014.

[31] Sandeep Tata and Jignesh M Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, 36(2):7–12, 2007.

[32] Vladimir Vapnik. *The nature of statistical learning theory.* Springer Science & Business Media, 2013.

[33] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural Computation*, 6(5):851–876, 1994.

[34] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[35] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):340–352, 2016.

[36] Weiren Yu and Julie Ann McCann. High quality graph-based similarity search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 83–92. ACM, 2015.

6 Appendix

6.1 Proof of Theorem 1

In this section, we show our main result (Theorem 1). We start from the definition of self bounding function [24].

Definition 1. *Let s_1, s_2, $\cdots$, s_k be independent random variables taking values from a set D. A function $f : D^k \to [0, +\infty]$ is a self bounding function if there exists a constant c and a function $g : D^{k-1} \to \mathbb{R}$ such that for any s_1, $\cdots$, s_{j-1}, s_{j+1}, $\cdots$, $s_k \in D$, the following conditions hold:*

$$0 \leq f(s_1, \cdots, s_k) - g(s_1, \cdots, s_{j-1}, s_{j+1}, \cdots, s_k) \leq c,$$

$$\sum_{j=1}^{k} [f(s_1, \cdots, s_k) - g(s_1, \cdots, s_{j-1}, s_{j+1}, \cdots, s_k)] \leq f(s_1, \cdots, s_k).$$

The following concentration inequality can be achieved for self bounding functions [7].

Lemma 1. *[7] If a function $Z = f(s_1, \cdots, s_k)$ is a self bounding function with constant c, then for $t \leq \mathbb{E}Z$,*

$$\Pr[\mathbb{E}Z - Z \geq t] \leq \exp\left(-\frac{t^2}{2c\mathbb{E}Z}\right).$$

For $t > \mathbb{E}Z$, the left probability is zero trivially. Here we take randomness over s_1, s_2, $\cdots$, s_k.

By using the above lemma, we can show a similar inequality for Rademacher average.

Lemma 2.

$$\Pr[\mathbb{E}R_S(F) \geq R_S(F) + t] \leq \exp\left(-\frac{kt^2}{4m\mathbb{E}R_S(F)}\right),$$

where $\mathbb{E}$ takes randomness over the samplings s_1, s_2, $\cdots$, s_k.

Proof. It suffices to show that $R_S(F)$ is a self bounding function with constant $c = 2m/k$. Define

$$Z = R_S(F) = \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i)\right],$$

$$G_j = \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i \neq j} \sigma_i f(s_i)\right].$$

It is clear that Z is non-negative:

$$Z \geq \sup_{f \in F} \left[\mathbb{E}_\sigma \frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i)\right] = 0.$$

Also it is clear that $Z \geq G_j$ for each j: suppose $\tilde{f}$ achieves the supreme of G_j. Then

$$\begin{aligned}
G_j &= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i) - \frac{2}{k} \sigma_j \tilde{f}(s_j)\right] \\
&= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i)\right] - \mathbb{E}_\sigma \left[\frac{2}{k} \sigma_j \tilde{f}(s_j)\right] \\
&= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i)\right] \leq Z.
\end{aligned}$$

Next we show $Z - G_j \le 2m/k = c$:

$$\begin{aligned} G_j &= \mathbb{E}_\sigma \sup_{f\in F} \left[\frac{2}{k}\sum_{i=1}^{k} \sigma_i f(s_i) - \frac{2}{k}\sigma_j f(s_j) \right] \\ &\ge \mathbb{E}_\sigma \sup_{f\in F} \left[\frac{2}{k}\sum_{i=1}^{k} \sigma_i f(s_i) \right] - \mathbb{E}_\sigma \sup_{f\in F} \left[\frac{2}{k}\sigma_j f(s_j) \right] \\ &\ge \mathbb{E}_\sigma \sup_{f\in F} \left[\frac{2}{k}\sum_{i=1}^{k} \sigma_i f(s_i) \right] - \frac{2m}{k}. \end{aligned}$$

Finally we need to verify $\sum_{j=1}^{k} Z - G_j \le Z$:

$$\begin{aligned} \sum_{j=1}^{k} G_j &= \mathbb{E}_\sigma \sum_{j=1}^{k} \sup_{f\in F} \left[\frac{2}{k}\sum_{i\ne j} \sigma_i f(s_i) \right] \\ &\ge \mathbb{E}_\sigma \sup_{f\in F} \left[\frac{2}{k}\sum_{j=1}^{k}\sum_{i\ne j} \sigma_i f(s_i) \right] \\ &= \frac{2(k-1)}{k}\mathbb{E}_\sigma \sup_{f\in F} \left[\sum_{j=1}^{k} \sigma_i f(s_i) \right] = (k-1)Z. \end{aligned}$$

□

We still need the following lemma on the relation between uniform deviation and Rademacher average.

Lemma 3.

$$\mathbb{E}\sup_{f\in F}[A_S(f) - A_D(f)] \le \mathbb{E}R_S(F),$$
$$\mathbb{E}\sup_{f\in F}[A_D(f) - A_F(f)] \le \mathbb{E}R_S(F).$$

Here we take randomness over the k samplings.

Proof. The proof idea is based on ghost samplings, i.e., independently draw another k samples: $s'_1, \cdots, s'_k$, and then we have

$$A_D(f) = \frac{1}{m}\sum_{i=1}^{m} f(i) = \mathbb{E}\frac{1}{k}\sum_{j=1}^{k} f(s'_j),$$

where $\mathbb{E}$ takes randomness over the k ghost samples. Thus

$$\begin{aligned} \mathbb{E}\sup_{f\in F}[A_S(f) - A_D(f)] &= \mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \mathbb{E}\frac{1}{k}\sum_{j=1}^{k} f(s'_j) \right] \\ &\le \mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \frac{1}{k}\sum_{j=1}^{k} f(s'_j) \right]. \end{aligned}$$

Since all the samples s, s' are independently identically distributed, flipping the sign of $f(s_i) - f(s'_i)$ will not change the expected supreme, i.e.,

$$\mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \frac{1}{k}\sum_{j=1}^{k} f(s'_j) \right] = \mathbb{E}\sup_{f\in F}\frac{1}{k}\sum_{i=1}^{k}\left[\sigma_i(f(s_i) - f(s'_i))\right],$$

where σ_i is uniformly distributed over $\{-1, 1\}$. Since

$$\mathbb{E} \sup_{f \in F} \frac{1}{k} \sum_{i=1}^{k} [\sigma_i (f(s_i) - f(s_i'))] \leq 2\mathbb{E} \sup_{f \in F} \frac{1}{k} \sum_{i=1}^{k} \sigma_i f(s_i) = \mathbb{E} R_S(F),$$

we have shown the first inequality. The second inequality is analogous. $\square$

We also need McDiarmid's inequality [22].

Lemma 4. *[22] Let $s_1, \cdots, s_k$ be independent random variables taking values from a set D. Suppose a function $h : D^k \to \mathbb{R}$ satisfies*

$$\sup_{x_1, \cdots, x_k, x_i' \in D} |h(x_1, \cdots, x_k) - h(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_k)| \leq c_i$$

for some constants c_i and every $1 \leq i \leq k$. Then for any $t > 0$, we have

$$\Pr[h(s_1, \cdots, s_k) - \mathbb{E}h(s_1, \cdots, s_k) \geq t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{k} c_i^2}\right).$$

By the above three lemmas, we can bound the difference between true average and sampled average as follows.

Lemma 5.

$$\begin{aligned} &\Pr\left[\sup_{f \in F} |A_D(f) - A_S(f)| \geq R_S(F) + t\right] \\ \leq &4 \exp\left(-\frac{2kt^2}{(m + \sqrt{8m\mathbb{E}R_S(F)})^2}\right). \end{aligned}$$

Proof. First by Lemma 3,

$$\begin{aligned} &\Pr\left[\sup_{f \in F} [A_D(f) - A_S(f)] \geq R_S(F) + t\right] \\ \leq &\Pr\left[\sup_{f \in F} [A_D(f) - A_S(f)] \geq \mathbb{E} \sup_{f \in F} [A_D(f) - A_S(f)] + at\right] \\ &+ \Pr\left[\mathbb{E}R_S(F) \geq R_S(F) + (1-a)t\right] \end{aligned}$$

for any $a \in [0, 1]$. Let

$$h(s_1, \cdots, s_k) = A_D(f) - A_S(f) = A_D(f) - \frac{1}{k} \sum_{i=1}^{k} f(s_i).$$

It is clear that

$$\begin{aligned} &\sup_{x_1, \cdots, x_k, x_i' \in D} |h(x_1, \cdots, x_k) - h(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_k)| \\ = &\sup_{x_1, \cdots, x_k, x_i' \in D} \left| \frac{1}{k} \sum_{j=1, j \neq i}^{k} f(x_j) + \frac{1}{k} f(x_i') - \frac{1}{k} \sum_{i=1}^{k} f(x_i) \right| \\ = &\sup_{x_1, \cdots, x_k, x_i' \in D} \left| \frac{1}{k} f(x_i') - \frac{1}{k} f(x_i) \right| \leq \frac{m}{k}. \end{aligned}$$

By McDiarmid's inequality,

$$\begin{aligned}&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq \mathbb{E}\sup_{f\in F}[A_D(f)-A_S(f)]+at\right]\\ \leq&\exp\left(-\frac{2a^2t^2}{\sum_{i=1}^k m^2/k^2}\right)=\exp\left(-\frac{2ka^2t^2}{m^2}\right).\end{aligned}$$

By Lemma 2,

$$\Pr\left[\mathbb{E}R_S(F)\geq R_S(F)+(1-a)t\right]\leq \exp\left(-\frac{k(1-a)^2t^2}{4m\mathbb{E}R_S(F)}\right).$$

Let $a=1/(1+\sqrt{8\mathbb{E}R_S(F)/m})$. Then putting everything together, we have

$$\begin{aligned}&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq R_S(F)+t\right]\\ \leq&2\exp\left(-\frac{2kt^2}{(m+\sqrt{8m\mathbb{E}R_S(F)})^2}\right).\end{aligned}$$

Similarly one can show

$$\begin{aligned}&\Pr\left[\sup_{f\in F}[A_S(f)-A_D(f)]\geq R_S(F)+t\right]\\ \leq&2\exp\left(-\frac{2kt^2}{(m+\sqrt{8m\mathbb{E}R_S(F)})^2}\right).\end{aligned}$$

Thus we have the inequality as desired. □

By the above lemma we have the following important corollary.

Corollary 1. *With probability at least $1-\delta$, we have*

$$\sup_{f\in F}|A_S(f)-A_D(f)|\leq R_S(F)+(m+\sqrt{8m\mathbb{E}R_S(F)})\sqrt{\frac{\log\frac{4}{\delta}}{2k}}.$$

We still need to upper bound $\mathbb{E}R_S(F)$. By Lemma 2, with probability at least $1-\delta$,

$$\mathbb{E}R_S(F)\leq R_S(F)+\sqrt{4m\mathbb{E}R_S(F)\frac{\log\frac{1}{\delta}}{k}}.$$

Or equivalently,

$$\sqrt{\mathbb{E}R_S(F)}\leq\sqrt{\frac{m}{k}\log\frac{1}{\delta}}+\sqrt{\frac{m}{k}\log\frac{1}{\delta}+R_S(F)}.$$

Hence with probability at least $1-2\delta$, we have

$$\sup_{f\in F}|A_S(f)-A_D(f)|\leq R_S(F)+\left(m+m\sqrt{\frac{8}{k}\log\frac{1}{\delta}}+m\sqrt{\frac{8}{k}\log\frac{1}{\delta}+\frac{8R_S(F)}{m}}\right)\sqrt{\frac{\log\frac{4}{\delta}}{2k}}.$$

We have shown Theorem 1.

6.2 Proof of Theorem 2

For any $s > 0$, by Jensen's inequality,

$$\begin{aligned}\exp(skR_S(F)) &= \exp\left(2s\mathbb{E}_\sigma \sup_{f\in F}\sum_{i=1}^k \sigma_i f(s_i)\right) \\ &\le \mathbb{E}_\sigma \exp\left(2s \sup_{f\in F}\sum_{i=1}^k \sigma_i f(s_i)\right) \\ &\le \mathbb{E}_\sigma \sum_{f\in F} \exp\left(2s \sum_{i=1}^k \sigma_i f(s_i)\right).\end{aligned}$$

By Hoeffding's Lemma [16],

$$\begin{aligned}&\mathbb{E}_\sigma \sum_{f\in F} \exp\left(2s \sum_{i=1}^k \sigma_i f(s_i)\right) \\ &\le \sum_{f\in F}\prod_{i=1}^k \exp\left(2s^2 f(s_i)^2\right) \\ &= \sum_{f\in F} \exp\left(2s^2 \sum_{i=1}^k f(s_i)^2\right).\end{aligned}$$

Let $\ell^2 = \sup_{f\in F}\sum_{i=1}^k f(s_i)^2$, and then

$$\sum_{f\in F} \exp\left(2s^2 \sum_{i=1}^k f(s_i)^2\right) \le |F| \exp\left(2s^2\ell^2\right).$$

Thus

$$R_S(F) \le \frac{1}{sk}(\log|F| + 2s^2\ell^2),$$

for any $s > 0$. It turns out that to minimize the right hand side of the above equation, we have

$$s = \sqrt{\frac{\log|F|}{2\ell^2}}.$$

Then

$$R_S(F) \le \frac{\ell}{k}\sqrt{8\log|F|}.$$

6.3 Discussions of Theorem 3

Theorem 3 can be treated as a special case of Theorem 1 when $m = c$. All the reasoning will not be affected and thus the desired bound follows.