Sketching Techniques for Collaborative Filtering

Yoram Bachrach
Microsoft Research Ltd.
Cambridge, UK
t-yobach@microsoft.com

Ely Porat
Bar-Ilan University
Ramat-Gan, Israel
porately@cs.biu.ac.il

Jeffrey S. Rosenschein
The Hebrew University
Jerusalem, Israel
jeff@cs.huji.ac.il

Abstract

Recommender systems attempt to highlight items that a target user is likely to find interesting. A common technique is to use collaborative filtering (CF), where multiple users share information so as to provide each with effective recommendations.

A key aspect of CF systems is finding users whose tastes accurately reflect the tastes of some target user. Typically, the system looks for other agents who have had experience with many of the items the target user has examined, and whose classification of these items has a strong correlation with the classifications of the target user. Since the universe of items may be enormous and huge data sets are involved, sophisticated methods must be used to quickly locate appropriate other agents.

We present a method for quickly determining the proportional intersection between the items that each of two users has examined, by sending and maintaining extremely concise "sketches" of the list of items. These sketches enable the approximation of the proportional intersection within a distance of ϵ, with a high probability of $1 - \delta$. Our sketching techniques are based on random min-wise independent hash functions, and use very little space and time, so they are well-suited for use in large-scale collaborative filtering systems.

1 Introduction

Recommender systems attempt to provide a user with recommendations regarding information items that she is likely to find interesting. Recommender systems have been applied to many domains, such as books, music, videos, images, web pages, and news. Such systems maintain a profile of each user, and compare it to certain reference characteristics. Sometimes these characteristics are obtained from the information item itself, in an approach called the *content based approach*, and sometimes from information kept regarding the tastes of other users, in the *collaborative filtering approach*.

Collaborative filtering (CF) makes predictions about the tastes of a user (filtering) through collaboration among multiple agents, each tied to a specific user (because each user has an agent, we will sometimes use the terms interchangeably). These CF systems predict whether a certain item is likely to interest the target user under the assumption that users whose past tastes are similar to the target user are likely to give a good prediction regarding the future tastes of the target user. Typically, CF systems operate in two steps. First, they seek users who share similar rating patterns with the target user. Second, they use the ratings from those like-minded users found in the first step to generate a prediction for the target user. Real-world CF systems need to handle huge volumes of information, as the universe of items is enormous, and there are many users.

Given a target user, we want to find another user who has rated many of the items the target user has rated. Also, it is likely that the prediction would be better if items rated by both users are a significant *proportion* of the items rated by either user. For example, it may be the case that two users have 100 songs that both have rated, but each of them has rated 10,000 songs. In this case, these 100 songs constitute only 1% of the songs each user has rated, so it harder to use the ratings of the first to understand the tastes of the second.

We propose methods for approximating the proportional intersection (PI) between the items that each of two users has rated. Our methods are very accurate, returning an approximately correct answer with very high probability, but require little information to be stored or sent. These methods thus provide a useful building block for CF systems. Our algorithms use *sketching techniques*, which construct an *extremely concise* representation of the items a user has rated, called a *sketch*. Rather than storing, transmitting, and processing the entire list of rated items, our algorithms operate on these extremely concise sketches, which are designed to enable a good approximation of the PI. Our algorithms use sketches based on random min-wise independent hash functions. We analyze the proposed methods via sampling techniques and Hoeffding's inequality to calculate the required sketch size given the desired accuracy and confidence for the PI's size. This analysis shows that very small sketch sizes are indeed sufficient to achieve high levels of accuracy and success probabilities.

2 Problem Statement

We now formally define the problem of computing the proportional intersection (PI) between two sets. We attempt to

model a simple collaborative filtering domain. In this case, each user of the CF system has a set of items she has rated. A certain user, Alice, asks the CF system to give a prediction for a certain item she has not examined. In order to provide Alice with a good prediction, we must first find users who have rated many items that Alice has also rated. It is also desirable to find users such that the number of items rated by both Alice and any of these users are a significant *proportion* of the items rated by either of them. We call the proportion of items rated by two users out of all items rated by each of them the *proportional intersection* (PI). After finding users with a high PI to Alice's items, the CF system could use the ratings of these users to build a prediction for Alice. A naive way to do this is to keep the entire list of rated items for each user on a central server. However, this is undesirable, as the universe of items is very large, and there are a huge number of users, each of whom has ratings for a significant number of items.

One solution for handling the size of the data sets is compressing the lists of rated items using some known compression method. We are interested in achieving a *much shorter* description of the items list than compression techniques could enable. Such a concise description is called a *sketch*. We aim to design a sketch that would allow us to discover the PI with target users, and that would be as small as possible. Thus, we are willing to accept some inaccuracy when finding users with a high PI, and we are willing to accept some probability of error. Given target accuracy $\epsilon > 0$ and a target confidence δ, our method must return an approximation $\hat{x}$ to the actual PI x, such that with probability of at least $1-\delta$ the approximation is accurate enough, so $|\hat{x}-x| \leq \epsilon$. Obviously, the size of the sketch would depend on both ϵ and δ, and we wish it to be as small as possible.

We now consider two users, Alice and Bob, and formally define proportional intersection and the sketching technique. We denote by C_1 the set of items Alice has rated, and by C_2 the set of items Bob has rated. These items are taken from a large universe U of items (U is several orders of magnitude larger then C_1 and C_2). For now, we will assume that $|C_1| = |C_2| = n$ (for example, the n items each user has rated most recently). We denote $C_1 \cap C_2 = T$.

Definition 1. *The proportional intersection (PI) of C_1 and C_2 is $\frac{|T|}{|C_1|} = \frac{|T|}{|C_2|}$.*

Under our sketching model, we only maintain the sketches of the rated item lists. Thus, all we have available are the sketches S_1 of C_1, and S_2 of C_2. A sketching framework is designed to allow for the approximation of the PI between any two users with a target confidence and accuracy. Let $C_1, C_2, \ldots, C_m$ be the rated item lists of the users, and $S_1, S_2, \ldots, S_m$ be their sketches. We currently assume that for any i, j we have $|C_i| = |C_j|$. Consider any two users, i and j, and denote the PI $p_{i,j} = \frac{|C_i \cap C_j|}{|C_i|} = \frac{|C_i \cap C_j|}{|C_j|}$.

Definition 2. *A PI sketching framework with confidence δ and accuracy ϵ maintains only $S_1, S_2, \ldots, S_m$, and for any two users, a_i and a_j, allows for the computation of $p_{i,j}$ with accuracy of at least ϵ and with confidence of at least δ. That is, the framework returns an approximation $\hat{p_{i,j}}$ to $p_{i,j}$ such that with probability of at least $1-\delta$ we have $|p_{i,j} - \hat{p_{i,j}}| \leq \epsilon$.*

The following sections suggest a technique that enables the building of a PI sketching framework.

3 Sketches for Approximating Proportional Intersection

Consider again Alice and Bob, with the set C_1 of items that Alice has rated, and the set C_2 of items that Bob has rated (from the universe U of items). Again, we denote $|C_1| = |C_2| = n$ and $C_1 \cap C_2 = T$, and denote $|T| = t$. A straightforward method of generating a sketch for the items of each user is simply sampling some r random items. This generates a sketch S_1^r for C_1 and S_1^r for C_2; each such sketch is just a set of r items. We can estimate $p_{1,2} = \frac{|C_1 \cap C_2|}{|C_1|}$ by computing $\hat{p_{1,2}} = \frac{|S_1^r \cap S_2^r|}{r}$. However, for the sketch to be small, we must use small values of r. The probability for each item from C_1 to appear in S_1^r is $\frac{r}{n-r+1}$. To note that a certain item is in both C_1 and C_2, it must appear in *both* S_1^r and S_2^r, and since r is small, this happens with very low probability.

When $r = 1$, for example, S_1^1 is simply a random item from C_1, and S_2^1 is a random item from C_2. In order to have an item in $S_1^1 \cap S_2^1$, the item chosen from C_1 to be in S_1^1 must be in T (out of the n items in C_1 there are only t such items), and the *same* item must be chosen from C_2 to be in S_2^1. Thus, the probability of having an item in $S_1^1 \cap S_2^1$ is $\frac{1}{t} \cdot \frac{1}{n}$, which is very small. Consider, however, the case where the sketch S_1 is the *minimal* item from C_1, and S_2 is the *minimal* item from C_2. If the minimal item in $C_1 \cup C_2$ is in T, we are guaranteed to have an item in $S_1 \cap S_2$. Of course, always using the *minimal* item would always generate the same S_1 and S_2, regardless of the PI that we are estimating. We overcome this by using min-wise independent functions.

Let H be a family of functions over the same source X and target Y, so each $h \in H$ is a function $h : X \to Y$, where Y is a completely ordered set. We say that H is min-wise independent if, when randomly choosing a function $h \in H$, for any subset $C \subseteq X$, any $x \in C$ has an equal probability of being the minimal after applying h.[1]

Definition 3. *H is a min-wise independent family, if for all $C \subseteq X$, for any $x \in C$,*

$$Pr_{h \in H}[h(x) = min_{a \in C} h(a)] = \frac{1}{|C|}.$$

[Indyk, 2001] shows that it is possible to construct min-wise independent families of hash functions such as the ones we require here, so from this point on we use these results.[2]

We now use integers to define the identity of items in U (where $|U| = u$), so any subset $C \subseteq U$ is simply a list of $|C|$ integers in $[u]$ (we use $[u]$ to denote $\{1, 2, \ldots, u\}$). Thus,

[1] We must choose h from H under a certain distribution. We will assume that h is chosen uniformly from H, although any distribution for choosing a member of h that would make H min-wise independent would do.

[2] Although we use Indyk's results, and assume it is possible to construct such families of min-wise independent hash functions, we do note that in fact for most practical applications it is possible to use any approximately uniform hash function, or even pairwise independent hash functions, and still obtain very good results.

Alice's rated items are $C_1 \subseteq [u]$, and Bob's rated items are $C_2 \subseteq [u]$. We continue to denote $|C_1| = |C_2| = n$ and $C_1 \cap C_2 = T$, and $|T| = t$.

Let H be a family of min-wise independent functions from $[u]$ to $[n^2]$ (so each $h \in H$ is a function $h : [u] \rightarrow [n^2]$). Note that the domain of each hash function in the family is an integer in the very large range of $[u]$, but the hashed values are in the much smaller range of $[n^2]$. The reason we use a range of n^2 integers rather than just n integers is to mitigate the effect of collisions in the hashed values.[3] The complete construction of min-wise independent families is fully explained in [Indyk, 2001], but for the purpose of understanding the current work it is only required to know that for any u and $n << u$ it is possible to construct a min-wise independent family of functions mapping from $[u]$ to $[n^2]$.

Let $h \in H$ be a randomly chosen function from H. We can apply h on all the integers in C_1 and examine the minimal integer we get, $m_1^h = min_{x \in C_1} h(x)$. We can do the same to C_2 and examine $m_2^h = min_{x \in C_2} h(x)$. We now compute the probability that $m_1 = m_2$: $Pr_{h \in H}[m_1^h = m_2^h] = Pr_{h \in H}[min_{x \in C_1} h(x) = min_{x \in C_2} h(x)]$.

Theorem 1. $Pr_{h \in H}[m_1^h = m_2^h] = \frac{p_{1,2}}{2 - p_{1,2}}$.

Proof. For brevity, we denote $p = p_{1,2}$. We use the size of the PI between Alice and Bob, $p = p_{1,2}$, to compute that probability. C_1 contains n items, $p \cdot n$ are items that are also present in C_2, and $(1-p) \cdot n$ that are not. C_2 contains n items, $p \cdot n$ are items that are present in C_1, and $(1-p) \cdot n$ that are not. In total $C_1 \cup C_2$ contains $p \cdot n + 2 \cdot (1-p) \cdot n = pn + (2-2p)n = n(2-p)$ items. When an item in T is minimal under h, i.e., for some $a \in T$ we have $h(a) = min_{x \in C_1 \cup C_2} h(x)$, we get that $min_{x \in C_1} h(x) = min_{x \in C_2} h(x)$. Since H is min-wise independent, we get that $Pr_{h \in H}[m_1^h = m_2^h] = \frac{pn}{n(2-p)} = \frac{p}{2-p}$. □

Building a PI sketching framework requires us to generate sketches that would allow us to get good estimates for the PI between any two users, $p = p_{i,j}$. We note that given $\alpha = Pr_{h \in H}[m_1^h = m_2^h]$ for Alice and Bob, we can easily obtain the PI between them, $p_{1,2}$. Since $\alpha = \frac{p_{1,2}}{2 - p_{1,2}}$, we get that $(2-p)\alpha = p$, so $p = 2\alpha - \alpha p$, and thus $p = \frac{2\alpha}{1+\alpha}$. Therefore, the problem of estimating the PI is reduced to estimating α.

Our suggested approximation algorithm relies on generating sketches based on a random sample of *several* such functions from the min-wise independent family H. The sketches are then used to approximate α, and thus approximate the PI.

[3][Indyk, 2001] shows how to construct an approximately min-wise independent family of hash functions where the source and target are the same space, i.e., each $h \in H$ is a function $h : [m] \rightarrow [m]$. To construct a min-wise independent family of functions from $[u]$ to $[n^2]$, we first apply an approximately uniform hash function $h' : [u] \rightarrow [n^2]$, and then apply the functions from the min-wise independent family from $[n^2]$ to $[n^2]$ which can be constructed using the methods in [Indyk, 2001]. Since $|C_i| = n$ but h' has a range of n^2, due to the Birthday Paradox principle, the probability of a collision (i.e., having $x \neq y \in C_i$ such that $h'(x) = h'(y)$) is very small, let alone a collision that occurs with the *specific* element we examine. For further details, see [Indyk, 2001].

We first define the sketches we use. Let $v_k = \langle h_1, h_2, \ldots, h_k \rangle$ be a tuple of k randomly chosen functions from the min-wise independent family H. Let C_i be the set of items ranked by agent a_i. We now define the sketch for C_i given v_k. We first denote the minimal item in C_i under h_j as $m_i^{h_j} = min_{x \in C_i} h_j(x)$.

Definition 4. *The H_k sketch of C_i, $S(C_i)$, is the list of minimal items in C_i under the k randomly chosen functions from h: $S^k(C_i) = (m_i^{h_1}, m_i^{h_2}, \ldots, m_i^{h_k})$.*

When k, the number of randomly chosen functions from H, is known, we simply write $S(C_i)$ rather than $S^k(C_i)$. We now show how to use the sketches of C_1 and C_2 to estimate the PI between them.

3.1 Estimating the Proportional Intersection, Given Sketches

We first note that due to Theorem 1, randomly choosing a function $h \in H$ and testing whether $m_1^h = m_2^h$ is in fact a Bernoulli trial, with a success probability of $\alpha = \frac{p_{1,2}}{2 - p_{1,2}}$. We define the random variable of this Bernoulli trial as:

$$X^h = \begin{cases} 1 & \text{if } m_1^h = m_2^h \\ 0 & \text{otherwise} \end{cases}$$

Given the sketches $S^k(C_1) = (m_1^{h_1}, m_1^{h_2}, \ldots, m_1^{h_k})$ and $S^k(C_2) = (m_2^{h_1}, m_2^{h_2}, \ldots, m_2^{h_k})$ we can easily test (for each $i \in \{1, \ldots, k\}$) whether $m_1^{h_i} = m_2^{h_i}$, and obtain the results of k such Bernoulli trials.

Lemma 1. *Let $h_1, \ldots, h_k$ be a set of k randomly-sampled functions from the min-wise independent family H, and $X_1 = X^{h_1}, \ldots, X_k = X^{h_k}$ be the series of k Bernoulli trials, as defined above. Let X be the number of successes in this series of Bernoulli trials, $X = \sum_{j=1}^{k} X_j$. Then the* maximum likelihood estimator *for $\alpha = \frac{p_{1,2}}{2 - p_{1,2}}$ is: $\hat{\alpha} = \frac{X}{k}$. This estimator is unbiased.*

Proof. Each such X_j is a single Bernoulli trial, and $Pr(X_j = 1) = \alpha$ and $Pr(X_j = 0) = 1 - \alpha$. $X_1, \ldots, X_k$ is a series of k such Bernoulli trials. X is the number of successes in this series of Bernoulli trials, $X = \sum_{j=1}^{k} X_j$, and thus has the binomial distribution $X \sim B(k, \alpha)$. Since the X_j's are independent (as H is min-wise independent) but identical Bernoulli trials, the *maximum likelihood estimator* for α is $\hat{\alpha} = \frac{X}{k}$. This estimator is known to be unbiased for the binomial distribution. □

The estimator $\hat{\alpha}$ by itself does not provide a bound on the probability that this value is approximately correct. Moreover, we are interested in approximating $p_{1,2}$, and not α. We begin by obtaining an estimate for α which is probably approximately correct (PAC) using the sample $X_1, \ldots, X_k$ of these k Bernoulli trials, and later handle deriving a probably approximately correct estimate for $p_{1,2}$. As mentioned in Definition 2, if for some $\epsilon > 0$, we consider values that are within a distance of ϵ from the correct value, $p_{1,2}$, accurate enough, and are willing to accept a certain low probability δ of having our estimator miss it by more than ϵ, we can derive

the appropriate number of functions k to use in the sketch. Obviously, larger values of k would increase our accuracy and confidence, but will of course increase the sketch size, which we want to minimize. The next section analyzes the sketch size, i.e., the required k, given the target confidence and accuracy.

4 Sketch Size Analysis

We now attempt to derive the necessary sketch size, k, that is required in order to build a PI sketching framework. We can formulate this problem as building a *confidence interval* for the PI, $p_{i,j}$, with an accuracy of ϵ and with confidence level of $1-\delta$. The interval we build is: $[\hat{p_{i,j}} - \epsilon, \hat{p_{i,j}} + \epsilon]$.

The interval is centered in $\hat{p_{i,j}}$, has a width of $2 \cdot \epsilon > 0$, and contains the true $p_{i,j}$ with a probability of at least $1-\delta$. To achieve the desired accuracy and confidence, the sketch size k must be long enough. To find the appropriate k, we must obtain an equation to tie together the required k, the confidence level δ, and the accuracy ϵ. Our analysis is based on Hoeffding's inequality [Hoeffding, 1963].

Theorem 2 (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables, where all X_i are bounded so that $X_i \in [a_i, b_i]$, and let $X = \sum_{i=1}^{n} X_i$. Then the following inequality holds.*

$$\Pr(|X - \mathrm{E}[X]| \geq n\epsilon) \leq 2\exp\left(-\frac{2\,n^2\,\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

We now use Hoeffding's inequality to derive the required sketch size for a desired confidence and accuracy regarding $\alpha = \frac{p}{2-p}$, and then show how to find the sketch size for a confidence and accuracy regarding p. Let $h_1, \ldots, h_k$ be a set of k randomly-sampled functions from the min-wise independent family H, and $X_1 = X^{h_1}, \ldots, X_k = X^{h_k}$ be the series k of Bernoulli trials, as defined above. Again, let $X = \sum_{j=1}^{k} X_j$, and take $\hat{\alpha} = \frac{X}{k}$ as an estimator for α. All X_i are either 0 or 1 (and are thus bounded between these values), and $\mathrm{E}[X] = k \cdot \alpha$. Thus, the following holds: $\Pr(|X - k\alpha| \geq k\epsilon) \leq 2e^{-2\,k\,\epsilon^2}$. Therefore the following also holds: $\Pr(|\hat{\alpha} - \alpha| \geq \epsilon) \leq 2e^{-2\,k\,\epsilon^2}$. We now compute the number of required samples in order to make sure that this probability is below some required confidence level δ.[4]

Theorem 3 (Sketch Size for Approximating α). *For any required accuracy $\epsilon > 0$ and required confidence level $1-\delta$, we can construct a confidence interval for α with width 2ϵ of the form: $[\hat{\alpha} - \epsilon, \hat{\alpha} + \epsilon]$.*

This interval holds the correct α with probability of at least $1-\delta$. The required sketch length (or equivalently, the required number of samples) to perform this is $k = \frac{\ln \frac{2}{\delta}}{2\,\epsilon^2}$. Equivalently, given k samples and a required confidence of $1-\delta$, the following is a confidence interval for α, with the required confidence level of $1-\delta$: $\left[\hat{\alpha} - \sqrt{\frac{1}{2k}\,\ln\frac{2}{\delta}}, \hat{\alpha} + \sqrt{\frac{1}{2k}\,\ln\frac{2}{\delta}}\right]$.

[4]Such computations are widely used for the analysis of randomized algorithms in various areas. For example, PAC learning algorithms are usually based on similar techniques.

Proof. We use Hoeffding's inequality to make sure the error does not exceed our target confidence level δ, and get: $Pr(|\hat{\alpha} - \alpha| \geq \epsilon) \leq 2\,e^{-2\,k\,\epsilon^2} \leq \delta$. We extract ϵ and k: $-2\,k\,\epsilon^2 \leq \ln\frac{\delta}{2}$. Equivalently: $\epsilon^2 \geq -\frac{\ln\frac{\delta}{2}}{2\,k}$. Finally we get the following equations: $\epsilon \geq \sqrt{\frac{1}{2k}\,\ln\frac{2}{\delta}}$ and $k \geq \frac{\ln\frac{2}{\delta}}{2\,\epsilon^2}$. □

The above sketch size analysis was for obtaining an approximation for α, and not for $p_{i,j}$. However, due to Theorem 1 we have $p_{i,j} = \frac{2\alpha}{1+\alpha}$. We are interested in building a PI sketching framework, with confidence $1-\delta$ and accuracy ϵ for $p_{i,j}$. We now show that by taking a sketch size that allows building a confidence interval for α with width of $\epsilon' = \frac{\epsilon}{3}$ and confidence δ, we obtained the required accuracy of ϵ and confidence of δ for $p_{i,j}$.

Theorem 4 (Sketch Size for Approximating $p_{i,j}$). *A PI sketching framework with accuracy ϵ and confidence δ can be obtained by using a sketch size of: $k = \frac{\ln\frac{2}{\delta}}{2\,\frac{\epsilon^2}{9}}$.*

Proof. For brevity we denote $p = p_{i,j}$. Due to Theorem 1, we have $p = \frac{2\alpha}{1+\alpha}$. Denote by k the sketch size required for obtaining an estimate $\hat{\alpha}$ for α, with accuracy of ϵ' and confidence $1-\delta$, as can be derived from Theorem 3. By using sketches of size k, we obtain an estimate $\hat{\alpha}$ such that with probability $1-\delta$ we have $\hat{\alpha} - \epsilon' < \alpha < \hat{\alpha} + \epsilon'$. We use the following as an estimator for p: $\hat{p} = \frac{2\hat{\alpha}}{1+\hat{\alpha}}$. We now analyze how good this estimator for p is. Since $\hat{\alpha} - \epsilon' \leq \alpha \leq \hat{\alpha} + \epsilon'$, we obtain the following equations: $2\hat{\alpha} - 2\epsilon' \leq 2\alpha \leq 2\hat{\alpha} + 2\epsilon'$ and $\hat{\alpha} - \epsilon' + 1 \leq \alpha + 1 \leq \hat{\alpha} + \epsilon' + 1$.

Thus: $\frac{2\hat{\alpha}-2\epsilon'}{\hat{\alpha}+\epsilon'+1} \leq \frac{2\alpha}{\alpha+1} = p \leq \frac{2\hat{\alpha}+2\epsilon'}{\hat{\alpha}-\epsilon'+1}$.

However, $\frac{2\hat{\alpha}-2\epsilon'}{\hat{\alpha}+\epsilon'+1} = \frac{2\hat{\alpha}}{\hat{\alpha}+\epsilon'+1} - \frac{2\epsilon'}{\hat{\alpha}+\epsilon'+1} > \frac{2\hat{\alpha}}{\hat{\alpha}+\epsilon'+1} - 2\epsilon'$.

We now examine the expression $\frac{2\hat{\alpha}}{\hat{\alpha}+\epsilon'+1}$:

$$\begin{aligned}
\frac{2\hat{\alpha}}{\hat{\alpha}+\epsilon'+1} &= \frac{2\hat{\alpha}}{\hat{\alpha}+1} \cdot \left(\frac{\hat{\alpha}+1}{\hat{\alpha}+\epsilon'+1}\right) \\
&= \hat{p} \cdot \left(\frac{\hat{\alpha}+1}{\hat{\alpha}+\epsilon'+1}\right) = \hat{p} \cdot \left(\frac{(\hat{\alpha}+1+\epsilon') - \epsilon'}{\hat{\alpha}+\epsilon'+1}\right) \\
&= \hat{p} \cdot \left(1 - \frac{\epsilon'}{\hat{\alpha}+\epsilon'+1}\right) = \hat{p} - \hat{p}\frac{\epsilon'}{\hat{\alpha}+\epsilon'+1} \\
&\geq \hat{p} - \frac{\epsilon'}{\hat{\alpha}+\epsilon'+1} \geq \hat{p} - \frac{\epsilon'}{1} = p - \epsilon'
\end{aligned}$$

Thus we have: $p \geq \frac{2\hat{\alpha}-2\epsilon'}{\hat{\alpha}+\epsilon'+1} \geq \frac{2\hat{\alpha}}{\hat{\alpha}+\epsilon'+1} - 2\epsilon' \geq \hat{p} - 3\epsilon'$. Similarly, we can show that: $p < \hat{p} + 3\epsilon$.

Note that we need only handle the case where $\epsilon' < \hat{\alpha}$. Our suggested method uses a sketch size and computes an estimation for the PI, $\hat{\alpha}$, using the sketches. The sketch size used is determined by the required approximation accuracy, as defined by ϵ (or ϵ'). If the computed $\hat{\alpha}$ (our estimation of the PI) is so small that $\hat{\alpha} < \epsilon'$, we can simply output $\hat{p} = \epsilon$ as our estimation (which for the required accuracy of ϵ would be accurate enough).

When $\epsilon' < \hat{\alpha}$ we have: $\frac{2\epsilon'}{\hat{\alpha}-\epsilon'+1} < 2\epsilon'$.

We also note that:[5]

$$\begin{aligned}\frac{2\hat{\alpha}}{\hat{\alpha}-\epsilon'+1} =& \frac{2\hat{\alpha}}{\hat{\alpha}+1}\cdot(\frac{\hat{\alpha}+1}{\hat{\alpha}-\epsilon'+1}) \\ =& \hat{p}\cdot(\frac{\hat{\alpha}+1}{\hat{\alpha}-\epsilon'+1}) = \hat{p}\cdot(\frac{(\hat{\alpha}+1-\epsilon')+\epsilon'}{\hat{\alpha}-\epsilon'+1}) \\ =& \hat{p}\cdot(1+\frac{\epsilon'}{\hat{\alpha}-\epsilon'+1}) = \hat{p}+\hat{p}\frac{\epsilon'}{\hat{\alpha}-\epsilon'+1} \\ \leq& \hat{p}+\frac{\epsilon'}{\hat{\alpha}-\epsilon'+1} \leq \hat{p}+\frac{\epsilon'}{1} = p+\epsilon'\end{aligned}$$

Earlier we have shown that: $p \leq \frac{2\hat{\alpha}+2\epsilon'}{\hat{\alpha}-\epsilon'+1} = \frac{2\hat{\alpha}}{\hat{\alpha}-\epsilon'+1} + \frac{2\epsilon'}{\hat{\alpha}-\epsilon'+1}$. Thus we obtain: $p \leq \frac{2\hat{\alpha}}{\hat{\alpha}-\epsilon'+1} + \frac{2\epsilon'}{\hat{\alpha}-\epsilon'+1} \leq \hat{p}+3\epsilon'$. We have now shown that: $\hat{p}-3\epsilon' \leq p \leq \hat{p}+3\epsilon'$. Thus, by using $\epsilon' = \frac{\epsilon}{3}$ in the formula of Theorem 3, we obtain the desired accuracy of ϵ for p, rather than for α. □

We now note that a similar approach, using similar transitions, can be used even if the set sizes of user items are similar but not identical. However, if these sets are vastly different in sizes, a different method must be used.

5 Applications in Collaborative Filtering Systems

We now examine the applicability of the above results to collaborative filtering systems. Our sketches require $k \log n$ bits per agent. Although k does not depend on the number of items each agent has ranked, n, it does depend on the desired confidence and accuracy.

Consider a recommender system for items in a large Internet music store, that holds over $|U| = 100{,}000{,}000$ music video clips. Each integer in that range requires 27 bits to represent. In such a system we may obtain direct rankings for clips by users, or infer them using the number of times a user has watched a clip, or the amount of time spent watching it. We assume each user has watched 10,000 clips.[6] The range of our hash functions is n^2, so we require $2 \log 10{,}000 = 27$ bits to represent each integer in this range. We use values of $\epsilon = 0.2$, and $\delta = 0.1$, which allow obtaining a moderately good estimation of the PI, with a pretty high probability. Using Theorem 4, we obtain the required sketch size of $k = 337$. Thus, a list of a user's clips requires $27 \cdot 10{,}000 = 270{,}000$ bits, whereas a sketch only requires $337 \cdot 27 = 9099$ bits. Since there are millions of users, this is a significant improvement. Another example is personalized web search.[7] Consider a recommender system for web pages, which suggests to users websites that were frequently visited by similar users. In this case, we deal with a much larger universe of 30,000,000,000 items,[8] so each item requires 35 bits to represent. Assuming that we have a list of 10,000 pages ranked per user (similarly to the previous example), a list representation would require 350,000 bits per user. Since the sketch size only depends on ϵ, δ and the number of items with which each user has interacted, the sketch size remains 9099 for this case as well, yielding an even more significant improvement.

[5] In the second-to-last transition we again use the fact that $\epsilon' < \alpha$.

[6] This is a large number, which might be appropriate for heavy users after several years of using the system, but it is still several orders of magnitude smaller than the total number of clips.

[7] We thank an anonymous reviewer for pointing out this interesting application.

6 Related Work

There are many examples of CF systems. An early example is Tapestry [Goldberg *et al.*, 1992], which relied on information manually inserted by users. Early systems with automated predictions were GroupLens [Resnick *et al.*, 1994] and Ringo [Shardan and Maes, 1995]. CF algorithms use correlations between human preferences to predict future preferences. [Resnick *et al.*, 1994] used the Pearson correlation coefficient, while [Shardan and Maes, 1995] examined several other measures.

This paper tackles the problem of handling the massive data sets in CF systems. A difficult step in generating a prediction for a target user is finding other users who share ranked items with the target user. We formulated this problem as computing the PI between the users' ranked items. Rather than maintaining the full lists of ranked items, we only maintain much shorter "sketches" that allow approximating the PI. Several papers deal with sketching. Massive data streams are fundamental to data processing applications; the increasing size of data sets has triggered the need for improved algorithms for processing huge inputs. Typically, papers that propose sketching techniques operate on large strings, and sketches are used to approximate various relations between strings. Much of that research considers the streaming model, where an algorithm can only make one pass over the data and, due to memory constraints, can only maintain a small sketch for further processing and queries. [Feigenbaum *et al.*, 2002] presents a sketch for approximating the L^1-difference between two streams, and [Cormode *et al.*, 2003] examines estimating the Hamming norm for massive data streams.

To the best of our knowledge, the current paper describes the first research that uses such sketching techniques to allow CF systems to handle massive data sets. Our methods are based on using a min-wise independent family of hash functions. Such functions have been used in the data stream model; families of this type have been studied in [Broder *et al.*, 2000; Mulmuley, 1996; Indyk, 2001]. Another application of such functions is estimating rarity of items and similarity over data stream windows [Datar and Muthukrishnan, 2002]. Our methods are similar to several other articles regarding Locally Sensitive Hashing (LSH) [Gionis *et al.*, 1999; Andoni and Indyk, 2008]. Our technique is an adaptation of the LSH framework for the specific case of CF systems. We focused on estimating PI, and provided bounds using current theoretical methods, based on assumptions regarding the CF scenario. There exist several other techniques of a similar nature. Random Projection methods, such as

[8] Measuring the exact number of webpages on the Internet is a very difficult task, however we believe 30 billion is in the right order of magnitude.

[Achlioptas, 2003] embed n points in a high-dimensional space into a k-dimensional space, where k logarithmically depends on n, while preserving pairwise distances up to a small factor. However, our technique is more tailored towards CF and computing the PI. Semantic Hashing [Salakhutdinov and Hinton, 2008] chooses codes for datapoints such that the Hamming distance between the codes correlates with the degree of semantic similarity between them. A similar approach, Spectral Hashing, is examined in [Weiss *et al.*, 2008], where it is shown that finding the optimal Semantic Hash code is NP-Hard. Although Spectral Hashing is computationally feasible, it still involves non-trivial computations of matrix thresholded eigenvectors of the graph Laplacian, whereas our approach is much simpler, and computationally efficient.

7 Conclusion

A major challenge in CF systems is the huge amount of information to be handled. A key building block of CF systems is finding users who share many ranked items. We formulated this problem as computing the PI, and suggested a sketching technique that allows doing so in an enormous universe of items, and with huge volumes of information regarding such rankings. Our method is based on concise sketches. To achieve a small sketch size, we sacrifice the exact computation of the PI, and only *approximate* it with a given accuracy and success probability. The sketch size is logarithmic in the desired confidence, and polynomial in the desired accuracy, remaining small even when the confidence and accuracy are high. Our method thus allows building scalable CF systems.

Much work remains open for future research. First, our methods only allow finding users with a high PI of ranked items with those of the target users. A CF system must also check for correlations between the users' rankings, and offer good recommendations based on this information. For a full solution, one must show how to tractably perform these steps for massive data sets. Second, we only analyzed the required sketch size theoretically. We believe empirical tests of our methods, using real data sets or using simulations, would allow us to improve the sketch size used, which could be of importance in real-world applications. Finally, we have focused on uses of our methods for CF systems, but we believe these methods are general. Future research could reveal other applications for these methods, for example in trust and reputation systems.

Acknowledgment

This research was partially supported by Israel Science Foundation grant #898/05.

References

[Achlioptas, 2003] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *JCSS: Journal of Computer and System Sciences*, 66, 2003.

[Andoni and Indyk, 2008] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[Broder *et al.*, 2000] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *JCSS: Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[Cormode *et al.*, 2003] Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using Hamming norms (how to zero in). *IEEE Trans. Knowl. Data Eng*, 15(3):529–540, 2003.

[Datar and Muthukrishnan, 2002] Mayur Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In Rolf H. Möhring and Rajeev Raman, editors, *Algorithms — ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17-21, 2002, Proceedings*, volume 2461 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.

[Feigenbaum *et al.*, 2002] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput*, 32(1):131–151, 2002.

[Gionis *et al.*, 1999] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB: International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers, 1999.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12), 1992.

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

[Indyk, 2001] Piotr Indyk. A small approximately min-wise independent family of hash functions. *Journal of Algorithms*, 38(1):84–90, January 2001.

[Mulmuley, 1996] Ketan Mulmuley. Randomized geometric algorithms and pseudorandom generators. *Algorithmica*, 16(4/5):450–463, 1996.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstorm, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[Salakhutdinov and Hinton, 2008] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, December 2008.

[Shardan and Maes, 1995] Upendra Shardan and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Using the Information of Others*, pages 210–217, 1995.

[Weiss *et al.*, 2008] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Processing Systems*, 2008.