2014 IEEE International Conference on Data Mining Workshop

Popular Items or Niche Items: Flexible Recommendation using Cosine Patterns

Yaqiong Wang*, Junjie Wu*, Zhiang Wu†, Hua Yuan‡ and Xu Zhang§
*School of Economics and Management, Beihang University, Beijing, China
†Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, China
‡School of Management, University of Electronic Science and Technology of China, Chengdu, China
§National Computer network Emergency Response Technical Team/Coordination Center of China, Beijing, China
Email: zhangxu_cncert@163.com

***Abstract*—Recent years have witnessed the explosive growth of recommender systems in various exciting application domains such as electronic commerce, social networking, and location-based services. A great many algorithms have been proposed to improve the *accuracy* of recommendation, but until recently the long tail problem rising from inadequate recommendation of niche items is recognized as a real challenge to a recommender. This is particularly true for ultra-massive online retailers who usually have tremendous niche goods for sale. In light of this, in this paper, we propose a pattern-based method called *CORE* for flexible recommendation of both popular and niche items. CORE has two notable features compared with various existing recommenders. First, it is superior to previous pattern-based methods by adopting cosine rather than frequent patterns for recommendation. This helps filter out spurious cross-support patterns harmful to recommendation. Second, compared with some benchmark methods such as SVD and LDA, CORE does well in niche item recommendation given particularly heavy tailed data sets. Indeed, the coupled configuration of the support and cosine measures enables CORE to switch freely between recommending popular and niche items. Experimental results on two benchmark data sets demonstrate the effectiveness of CORE especially in long tail recommendation. To our best knowledge, CORE is among the earliest recommenders designed purposefully for flexible recommendation of both head and tail items.**

***Index Terms*—Recommender System; Long Tail; Cosine Pattern; Collaborative Filtering**

I. Introduction

The explosive growth and variety of information available on the Web frequently overwhelm people. Since the introduction in the mid-1990s [16], recommender systems have proved to be a promising means for coping with information overload, assisting people in making better decisions. Various recommendation methods have been proposed successively to gain higher recommendation accuracy. Among them there are relatively simple but direct methods including content-based methods [4], pattern-based methods [8], and other early CF(collaborative filtering)-based methods [29], some of which have been playing an important role in highly rated Internet sites such as amazon.com and movielens.com. Later on, some relatively complex techniques like matrix factorization [18] and probabilistic topic modeling [7], [20] were also introduced to build recommender systems, and those model-based methods [5] thus gained increasing interests from researchers of various domains. In recent years, with the prosperity of online social media and ubiquitous environment, rich heterogeneous information began to be integrated into traditional models for enhancing recommendation accuracy. This brings context-aware recommendation [2], social recommendation [12], mobile recommendation [24], and so on.

It is not until recently that some researches began to investigate the long tail phenomenon in recommendation. This brings heated discussions on the nature of a recommender in terms of both accuracy and diversity of recommended items. For instance, [3] pointed out that niche items could grow to larger share of total sales, and thus should not be ignored as by most classic methods. Also, more sales of long tail items could be beneficial to customers [6], [11] by helping them discover items she/he did not know before. Some researches have come up with initial solutions, e.g., clustering tail items in advance [23], re-ranking candidate items [1], and combining with graph based algorithms [34], to promote long-tail items recommendation. These studies, however, are still in the initial stage, and most of them lack sufficient flexibility in recommending both hot and cold items to meet different application scenarios.

In light of this, we propose in this paper a pattern-based method called *CORE* for flexible recommendation of both popular and niche items. Our main contributions are summarized as follows:

- We revisit the long-standing pattern-based recommendation methods and give in-depth analysis on their bottlenecks. We reveal that frequent patterns or association rules are subject to the limitation of the support measure and easy to include spurious patterns, which would eventually damage recommendation accuracy.
- We propose CORE, a new recommendation method based on cosine patterns instead of frequent patterns. In this way, CORE can generate more accurate recommendations without the influence of spurious cross-support patterns [33]. More importantly, CORE is particularly suitable to long-tail recommendation, and has flexibility in recommending both popular and niche items.
- We conduct extensive experiments on two benchmark data sets to demonstrate the superiority of CORE to existing pattern-based methods as well as some classic CF-based methods on long tail recommendation. To our best

978-1-4799-4274-9/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDMW.2014.157

knowledge, CORE is among the earliest recommenders designed purposefully for flexible recommendation.

The remainder of this paper is organized as follows: In Section II, we introduce cosine patterns and illustrate why frequent patterns are harmful to recommendation. In Section III, we present CORE with details and discuss its appealing properties. Experimental results are given in Section IV. In Section V, we introduce related work and finally conclude our work in Section VI.

II. Preliminaries

In this section, we give some background knowledge on cosine patterns. We assume readers are familiar with frequent patterns, and compare the two patterns for recommendation.

A. Definition of Cosine Pattern

In associative analysis, a transaction T_j is a set of items $\{i_1, i_2, \cdots, i_{|T_j|}\}$ purchased by a user u_j. A pattern, in the form of $\{i_1, i_2, \cdots, i_K\}$, is a K-itemset I derived from a transaction data set $\mathcal{T}$. The *support count* of I is defined as $\sigma(I) = |\{T_j | I \in T_j, T_j \in \mathcal{T}\}|$, i.e., the number of transactions containing I. The *support* of I is then given by $s(I) = \sigma(I)/|\mathcal{T}|$, and the *cosine* of I is defined as $\cos(I) = s(I)/\sqrt[K]{\prod_{k=1}^{K} s(\{i_k\})}$ [32]. Note that when $K = 2$, $\cos(I)$ reduces to the cosine similarity between two binary vectors V_1 and V_2, with $V_{kj} = 1$ if $i_k \in T_j$ and 0 otherwise.

As defined in [32], an itemset I is called a *cosine pattern* with respect to τ_s and τ_c, if $s(I) \geq \tau_s$ and $\cos(I) \geq \tau_c$, where $\tau_s, \tau_c \in [0, 1]$ are the minimum thresholds of *support* and *cosine*, respectively. Apparently, a cosine pattern differs from a frequent pattern by taking both τ_s and τ_c as thresholds for pattern validation. A higher τ_s but lower τ_c would result in cosine patterns more similar to freqent patterns (containing more frequent items). But in contrast, a lower τ_s but higher τ_c could lead to rare but truly cohesive patterns (containing more niche items) . In a nutshell, the coupled configuration of τ_s and τ_c brings extra flexibility to cosine patterns in covering both popular and niche items. This lays the foundation of CORE for flexible recommendation.

B. Property of Cosine Pattern

While the *support* of a pattern measures the significance of correlation among items, the *cosine* measures the degree of *cohesiveness* among items. An intuitional understanding of this is: if the items of I always appear concurrently in transactions, i.e., with the highest cohesiveness, $\cos(I) = 1$. To further illustrate this, in what follows, we introduce an important property of cosine.

In real-world applications, many transaction data sets, especially those with many popular or common items, have inherently skewed support distributions which often lead to the so-called "cross-support patterns" (CSP) [33] defined as follows:

Definition 1: Given $\theta \in (0, 1)$, I is a cross-support pattern w.r.t. θ if $\frac{\min_{i_j \in I} s(\{i_j\})}{\max_{i_j \in I} s(\{i_j\})} < \theta$.

According to the definition, a CSP typically represents spurious associations among items with substantially different *support* levels. They often appear as frequent patterns when a relatively small *support* threshold is set for frequent pattern mining, and hence may lead to inaccurate recommendations in a pattern-based recommender. One solution is to set a high *support* threshold, but it will lead to the omission of niche items in frequent patterns, and finally degrade the diversity of a pattern-based recommender. This illustrates why a frequent pattern-based recommender cannot perform as well as some model-based methods.

Interestingly, it has been proved in Theorem 1 of [32] that the cosine measure has the *anti-CSP property*. That is, a cross-support pattern tends to have a small *cosine* value and therefore cannot pass the threshold as a cosine pattern. In other words, cosine patterns are more robust than frequent patterns in resisting cross-support patterns. This partially explains why using cosine patterns rather than frequent patterns for pattern-based recommendation would be beneficial.

C. Cosine Pattern Mining

Mining of cosine patterns is not a trivial task. Fortunately, it has been found in [32] that the *cosine* measure possesses the so-called *conditional anti-monotone property* (CAMP), a property similar to the anti-monotone property of *support*. CAMP can be used to establish a quasi-Apriori Principle for cosine pattern mining as follows:

Theorem 1: Assume I_2 is an immediate superset of I_1, and for $i = I_2 \setminus I_1$, $s(\{i\}) \geq s(\{i_k\}) \ \forall \ i_k \in I_1$. If I_1 is not a cosine pattern, then I_2 will not be either.

The proof is not very hard by noting that the geometric average of the support values of all the items in I_2 is no less than that of I_1. This quasi-Apriori Principle can be used to design efficient algorithms for cosine pattern mining. In the previous work [32], we designed an Apriori-like algorithm called *CosMiner* to find cosine patterns using a breadth-first traversal strategy. The key point is, the $\mathcal{F}_k \times \mathcal{F}_k$ strategy for candidate generation is no longer valid given the CAMP of the cosine measure. Later in [31], we implemented a FP-growth-like algorithm called *CoPaMi* to mine cosine patterns. This depth-first traversal method is easier to coordinate with CAMP, and thus shows great potential in big data computing.

III. CORE: Cosine Pattern-based Recommender

This section describes CORE, a cosine pattern-based recommender for flexible recommendation. We first introduce how to use cosine patterns to generate recommendations. We then illustrate how CORE can be refined for efficient recommendation. We finally discuss how CORE could flexibly recommend both popular and niche items by configuring the *cosine* and *support* measures.

A. Naïve CORE

Here, we introduce the basic recommendation scheme of CORE, i.e., *naïve CORE* for short. It consists of three main phases as follows:

Algorithm 1 Recommendation based on Cosine Patterns

1: **procedure** REC($\mathcal{T}, CP, K$)
2: **for** each transaction $T_j \in \mathcal{T}$ **do**
3: Identify the set of target items I_t;
4: **for** each item $i_t \in I_t$ **do**
5: Identify CP_t from CP; ▷ cosine patterns covering i_t
6: Identify RB_t from CP_t; ▷ recommendation base
7: $score(i_t) = \sum_{I \in RB_t} \cos(I)$;
8: **end for**
9: Sort the items in I_t by $score$;
10: Generate recommendation list L_j by selecting top-K items;
11: **end for**
12: **return** $\mathcal{L} = \bigcup_j L_j$;
13: **end procedure**

Transactions Generation. In real-world applications, there are basically two types of data available for recommender systems, i.e., explicit ratings (e.g., scores, stars) and implicit ratings (e.g., browses, clicks, bookmarks). Data in the implicit form are natually transactions. But for data in the explicit form, we should map real-value ratings into binary values: rated (1) and unrated (0), with the assumption that rated items are users' interests regardless of the rating values.

Pattern Generation. We run CosMiner or CoPaMi on transaction data to get cosine patterns. The *support* and *cosine* thresholds should be carefully set to meet data characteristics and recommendation preferences. Typically, a relatively large *support* threshold will result in cosine patterns covering more frequent/popular items. However, if more rare/niche items are expected in recommendation, we should lower the *support* threshold but set a relatively large *cosine* threshold to filter spurious cross-support patterns. The coupled configuration of *support* and *cosine* indeed can bring us more flexibilities in recommendation. Note that all the cosine patterns are mined off-line and saved for the subsequent recommendation.

Recommendation. In this phase, CORE generates a recommendation list for each user based on his/her rated items and pertinent cosine patterns. Let T_j be the transacion containing rated items of user u_j, and let i_t be a target item to be recommended to u_j, $i_t \notin T_j$. Denote the set of cosine patterns containing i_t as CP_t. CORE then traverses each pattern $I \in CP_t$ to see whether $I \setminus \{i_t\} \subseteq T_j$. If so, we include I into the recommendation base RB_t, i.e., $RB_t = RB_t \bigcup I$. The score of i_t is then given by $score(i_t) = \sum_{I \in RB_t} \cos(I)$. We finally rank all the target items to be recommended to u_j by their scores and generate recommendation list containing top-K items. Algorithm 1 shows the pseudocodes describing the recommendation process.

B. Refined CORE

The Naïve CORE mentioned above is merely for the better understanding of the recommendation process. In what follows, we investigate how to refine it to achieve higher efficiency. Our focus is on the third phase of CORE.

In Algorithm 1, two inefficient points should be further addressed. The first is the loop for each target item i_t (see Line 4), which means multiple traversals of all the cosine patterns. The other is the exhaustive matching of patterns with a user's transaction (see Line 6), which could be very time-consuming. To deal with these, we propose a special structure called *Mined-Pattern Tree* (MP-tree) for compact representation of cosine patterns. Given a user transaction, we then traverse the MP-tree only once to find all recommendation bases for every target item simultaneously. In what follows, we will show how to construct the MP-tree and why traversing MP-tree for recommendations is efficient.

Definition 2 (MP-tree): MP-tree is a multiple tree structure defined below: (1) It consists of one root labeled as "$null$" and a set of nodes as the children of the root; (2) Each node has two fields: *item_name* and *pattern_value*, but for the children of the root the second field is void.

To illustrate the construction of MP-tree, we take the pattern set in Fig. 1 as an example. First of all, we sort items in each cosine pattern in a *support-ascending* order. Initially, the MP-tree contains only the root node. It then grows in the following way:

1) After reading the first pattern $\{A, B\}$, the nodes labeled as A and B are created. The path $null \rightarrow A \rightarrow B$ is then added. The value of $\cos(\{A, B\})$ is stored in the "pattern_value" field of the end node B, as shown in Fig. 1(a).

2) The second pattern $\{A, B, C\}$, shares a common prefix $\{A, B\}$ with the first pattern. As a result, a new node marked as "$C : \cos(\{A, B, C\})$" is added to the end of the path $null \rightarrow A \rightarrow B$.

3) This process continues until every pattern has been mapped onto one of the paths in the MP-tree. The resulted MP-tree is shown in Fig. 1(c).

In Naïve CORE, for a user u_j the target items set I_t is derived from the difference set of universal set of all items and his or her transaction T_j. This I_t is actually redundant since some items in it might never exist in any mined patterns, which would cause unnecessary traversals of all patterns for RB_t. With the MP-tree, the identification of target items set I_t could be embedded in the process of traversing T_j and MP-tree, so we could identify each i_t, its RB_t and $score(i_t)$ with only single traversal. This task could be transformed to traversing T_j and searching the MP-tree for i_t-targeted PATHs. The i_t-targeted PATH is defined as follows:

Definition 3 (i_t-targeted PATH): Let $P_t = \{i_1, \cdots, i_p\}$, $\forall 1 \leq j < p, i_{j+1} \in child\_list[i_j]$ denote a path in the MP-tree. P_t is a i_t-targeted PATH, if (1) the start node i_1 is at the second level of the MP-tree, i.e., $i_1 \in child\_list[root]$; (2) $i_t \in P_t, i_t \notin T_j, \forall i_j \in P_t \setminus \{i_t\}, i_j \in T_j$.

The i_t-targeted PATHs actually correspond to all cosine patterns that would contribute to $score(i_t)$, thus each i_t could be identified along the way of the *depth-first* traversal of the multiple tree. It is worth noting that the traversal of each path of MP-tree could stop once the second node that is not in T_j is detected, which again avoids unnecessary checking of longer patterns. Due to the limited space, we omit the details of the searching algorithm here.

Take the user transaction T_1 and MP-tree in Fig. 1 as an example. After reading the first item A in the given transaction, our algorithm first extracts the subtree with A as the root node

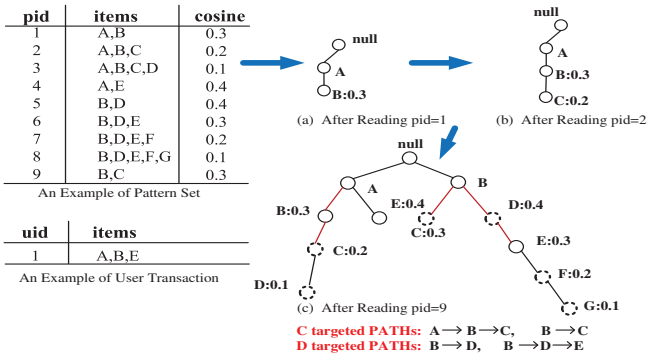


Fig. 1: Illustration of MP-tree and i_t-targeted PATHs

and then traverses along the path "$A \rightarrow B \rightarrow C \rightarrow D$". When the algorithm traverses to node C, the PATH "$A \rightarrow B \rightarrow C$" would be identified as a C-targeted PATH because it satisfies Definition 3, i.e., the first item A is at the second level of MP-tree, and all items except C are in T_1. Then, the second field of node C will be added to $score(C)$. Similarly, after reading the second item B in the transaction, another C-targeted PATH "$B \rightarrow C$" would be found and again $cos(\{B, C\}) = 0.3$ will be added to $score(C)$. Next, on the other branch, D-targeted PATHs "$B \rightarrow D$" and "$B \rightarrow D \rightarrow E$" would be found and then $cos(\{B, D\}) = 0.4$, $cos(\{B, D, E\}) = 0.3$ would be added to $score(D)$. Specially, traversal of this branch would stop at node F since two items that are not in T_1, namely, D and F, are detected, so there is no need to traverse down to check longer path "$B \rightarrow D \rightarrow E \rightarrow F \rightarrow G$". From this example we can see that using constructed MP-tree, it is much faster to calculate the recommendation score of each candidate item for each user.

C. Long Tail Recommendation

For real world practices, recommender systems based on cosine interesting pattern mining are flexible in recommending items at different levels of popularity. When *min_supp*, e.g., τ_s is sufficiently large, the pruning effect of *cosine* would be weakened and only highly frequent patterns would be used for recommendation. That is, the system would mostly recommend popular items just like traditional collaborative filtering methods. Then by gradually reducing *support* threshold τ_s, we could make sure the rare but interesting patterns be reserved. In this case, tail items that are not so popular could be recommended. If the recommendation performance is not so satisfying yet, considering some data may have longer tails and thus are even sparser, *cosine* threshold τ_c could be further reduced slowly to add in more interesting patterns. In the following Section IV-D, we could see clearly the trend of the ratio of recommended niche items under different settings of *support* and *cosine*.

Actually, what drives us most to consider applying cosine interesting patterns to recommender systems is the desirable feature of *cosine* to disclose truly associated patterns. As mentioned earlier, a pattern with low *support* would be missed in frequent pattern or association rule mining process, but this pattern could still be a high *cosine* pattern as long as items in it always appear in the same transaction. What's more intriguing is that this kind of patterns(low *support* but high *cosine*) correspond to the niche itemsets in business scenario which should not be discarded or ignored. Low *support* cosine patterns could catch the relation among tail items. This kind of "cold relation", commonly existing in data with more significant long tail property, is important for boosting tail items recommendations, but it is rather difficult to find by traditional recommender systems. In the following Section IV-D, we could see that on data with longer tail, CORE performs better than classic collaborative filtering methods on both accuracy and the ability to recommend niche items.

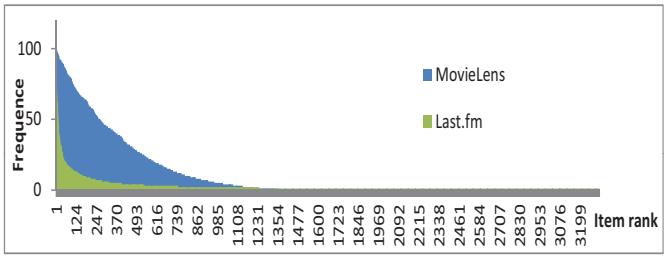


Fig. 2: Rating Distribution of Data Sets ($P(X \geq a)$)

TABLE I: Statistical Description of Data Sets

	Scale	#Users	#Items	#Ratings	Sparsity(%)
MovieLens	[1,5]	943	1682	100,000	93.70
Last.fm(Music)	[1,5]	635	4100	14175	99.46

Sparsity=100(1−#Ratings/(#Users×#Items)).

IV. Experimental Results

In this section, we conduct extensive experiments to evaluate CORE. Specifically, we first show the comparative results of CORE against two types of baseline methods, i.e., the pattern-based and the classic CF-based methods. We then analyze the advantages of CORE in challenging the long-tail recommendation problem.

A. Experimental Setup

CORE is tested on two widely used data sets: `MovieLens` and `Last.fm`, publicly available at: http://grouplens.org/datasets/hetrec-2011/. For `Last.fm`, we choose users with at least ten ratings to avoid extreme sparsity. Fig. 2 shows the rating distribution over items, where the long-tail effect is far more significant for `Last.fm` than for `MovieLens`. Table I lists some statistics of the two data sets.

We compare CORE with two types of benchmark methods. The first category consists of association rule-based method (AR) [21] and frequent pattern-based method (FP). The second one include three widely adopted collaborative filtering methods, i.e., UCF [25], SVD [27] and LDA [30]. When comparing with pattern-based methods, we follow Ref. [21] and adopt five-fold cross validation method, which splits users into collaborative users (for pattern generation) and target users (for evaluation). We then erases some percentage of the ratings of each target user to form the test set. When

TABLE II: Pattern Comparison

MovieLens	*support*(%)	#pattern	#items_covered
AR	20	528331	393
FP	12	806568	431
CORE	3	180099	899

confidence=0.9 for AR; *cosine*=0.4 for CORE.

comparing with CF methods, however, we adopt the validation method used in most literature pertaining to recommender systems. That is, we randomly erase some percentage of the ratings of each user for the training purpose, and the erased ratings are then used for testing. The percentage is varied from 10% to 90% to test the robustness of the algorithms in different sparsity levels.

We use *recall*, *precision* and *f-measure* to evaluate the accuracy of recommendation. For each user U_i,

$$\text{recall}_{U_i} = \frac{\#hits_i}{|T_i|}, \ \text{precision}_{U_i} = \frac{\#hits_i}{K}, \tag{1}$$

$$\text{f-measure}_{U_i} = \frac{2 \times \text{recall}_{U_i} \times \text{precision}_{U_i}}{\text{recall}_{U_i} + \text{precision}_{U_i}}, \tag{2}$$

in which $\#hits_i$ is the number of items in the recommendation list that are also in U_i's test set, $|T_i|$ is the number of test ratings of U_i, and K is the fixed length of the recommendation list, with a default value 10. The overall accuracy can then be obtained by averaging the measures on all the testing users.

B. Comparison with Pattern-based Methods in Accuracy

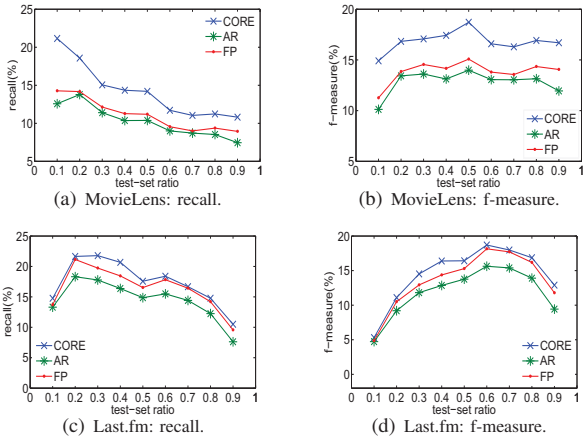
We here demonstrate the superiority of CORE to AR and FP. We record the best results of each algorithm by carefully tuning their parameters, i.e., *support* for AR, FP and CORE, *confidence* for AR, and *cosine* for CORE. Fig. 3 and Fig. 4 show the comparative results.

As can be seen from Fig. 3, in terms of *f-measure*, CORE outperforms AR and FP no matter what test-set ratio is set. A closer look at the subfigures reveals that the advantage of CORE is more prominent on `MovieLens` rather than `Last.fm`. This could be explained by the less sparsity of `MovieLens` (see Table I), which is prone to generate more patterns during frequent pattern mining. Therefore, to avoid the over-growth of patterns or rules, we have to set relatively higher *support* thresholds for FP and AR, which lead to huge volume of similar patterns or rules but smaller coverage of less frequent items. In contrast, CORE can set a much lower *support* threshold and rely on the *cosine* measure to discard cross-support patterns, which helps avoid missing some truly interesting patterns consisting of less frequent items. Table II illustrates this nicely.

Fig. 4 shows the sensitivity of recommendation precision to K, the length of recommendation list (test-set ratio = 30%). As can be seen, CORE again beats the other two pattern-based methods in terms of the *precision* measure, and the advantage seems very stable under different configurations of K.

C. Comparison with Classic CF-based Methods in Accuracy

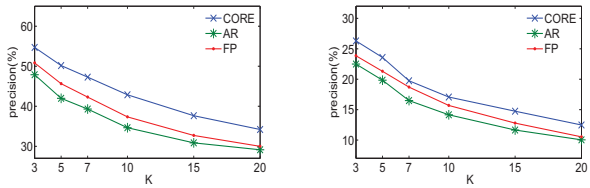
In this section, we compare the performance of CORE with classical collaborative filtering methods including UCF, SVD and LDA. In each group of experiments, we record the best results of all methods by tuning their parameters, namely, the number of neighbors N for UCF, the number of dimensions D for SVD, the number of latent factors F for LDA, and the *support* and *cosine* measures for CORE. Fig. 5 and Fig. 6 show the performances of all approaches with respect to different test-set ratios and different lengths of recommendation lists.



(a) MovieLens: recall. (b) MovieLens: f-measure.

(c) Last.fm: recall. (d) Last.fm: f-measure.

Fig. 3: CORE vs. AR/FP on Different Test-Set Ratios



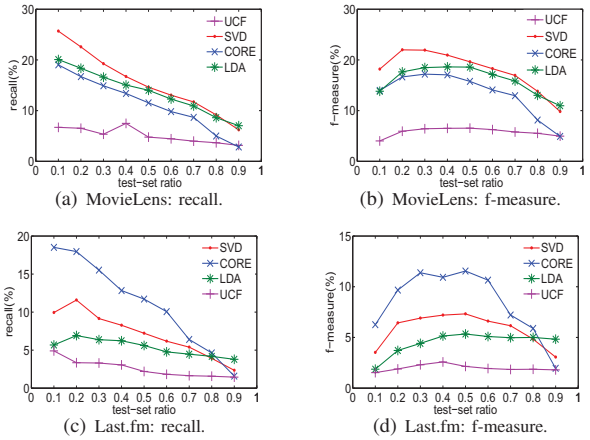
(a) MovieLens: precision. (b) Last.fm: precision.

Fig. 4: CORE vs. AR/FP on Different K's

From Fig. 5 we can see the opposite performances of CORE against baselines on two data sets. That is, for `MovieLens`, SVD and LDA outperform CORE with the gaps of 4% and 2.3%, respectively, on average, although UCF performs the worst among all the recommenders. As for `Last.fm`, the situation is quite opposite, CORE performs remarkably the best on almost all parameter configurations, and the gaps on average in terms of *recall* are 4%, 6%, 8% for SVD, LDA and UCF, respectively.

Fig. 6 further shows the recommendation quality of all methods on different lengths of recommendation list. The setup of this group of experiments is similar to the one given in Section IV-B. The results again illustrate that while SVD performs the best on `MovieLens`, CORE performs much better than the benchmarks on `Last.fm` in most cases.

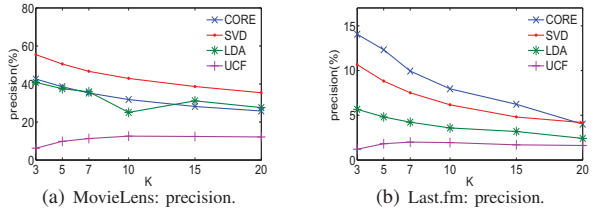
To sum up, CORE shows interesting characteristics in recommendation. On one hand, it dominates existing pattern-based methods by adopting the *cosine* measure. On the other, it shows opposite performances when comparing with the CF-based baselines on two data sets. In what follows, we relate the second point with the long tail phenomenon of items and unveil the merits on CORE in niche item recommendation.



(a) MovieLens: recall. (b) MovieLens: f-measure.

(c) Last.fm: recall. (d) Last.fm: f-measure.

Fig. 5: CORE vs. SVD/LDA/UCF on Different Test-Sets



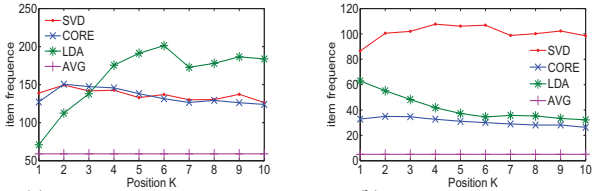
(a) MovieLens: precision. (b) Last.fm: precision.

Fig. 6: CORE vs. SVD/LDA/UCF on Different K's

D. Beyond Accuracy

In this section, we demonstrate the merit of CORE in flexible recommendation. Specifically, we show how CORE can switch freely from recommending tail items to recommending head items. This, to our best knowledge, is a novel contribution to long tail recommendation.

Comparison of item popularity. Fig. 7 shows the average popularity of recommended items at each position of recommendation list (test-set ratio=30%), which is measured by item frequency averaged on all users. From the figure we can see that while SVD and CORE tend to recommend items with similar popularity on `MovieLens`, they differ from each other drastically on `Last.fm`. That is, the top-10 items recommended by CORE are much less popular than the items recommended by SVD. In Table III, we also show the average ratio of niche items (here we mean items rated less than the average) in each user's recommendation list. The result agrees well with the observation from Fig. 7.



(a) MovieLens: item frequence. (b) Last.fm: item frequence.

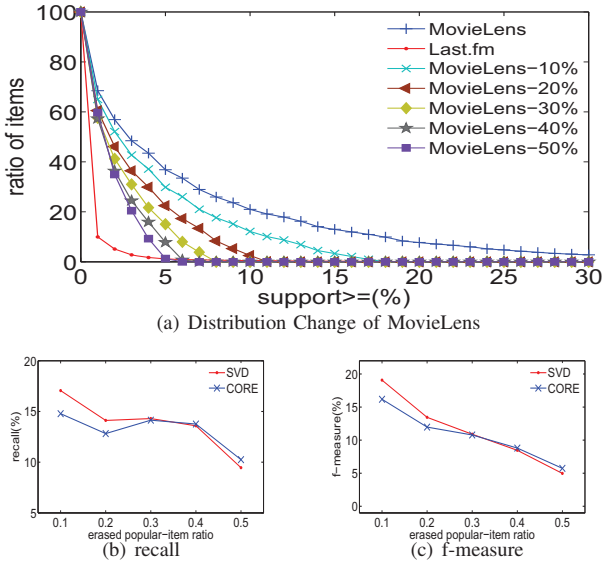
Fig. 7: Average Popularity of Recommended Items

TABLE III: Ratio of Niche Items Recommended

Data Set	AvgFreq.	SVD(%)	LDA(%)	CORE(%)
MovieLens	59	26.24	15.6	25.35
Last.fm	5	0.3	5.1	16.9

Generally speaking, popular items are more easier to be recommended by SVD, because such matrix factorization model tends to retain principal components while discarding niche features. Moreover, CORE fails to recommend as many niche items on `MovieLens` as on `Last.fm`. This is due to the fact that denser data like `MovieLens` usually need a higher *support* threshold to avoid pattern over-growth, which weakens the function of the *cosine* threshold taken by CORE to find less frequent but really interesting patterns. To illustrate this, we should notice that the best recommendation by CORE on `MovieLens` is obtained with $support = 1\%$, and the number reduces to 0.3% on `Last.fm`.

Preference to Longer Tail. Observing the recommendation list analysis above and accuracy comparison in Section IV-C, we can see that CORE has an edge over SVD in both accuracy and the specialty to find niche items on data like `Last.fm`. From Fig. 2 showed earlier, we see that while all data have long-tail property, `Last.fm` has much longer tail. We therefore guess that on data with longer tail, CORE is capable of recommending niche items without sacrificing much accuracy. To confirm this idea, we change the rating distribution of `MovieLens`, hoping to intensify its long tail problem by leaving out some popular items. Specifically, we first rank items by their *support*, then delete items ranking in top 10%, 20%, 30%, 40% and 50%, successively. Distributions of new data generated from original `MovieLens` is shown in Fig. 8 (a).



(a) Distribution Change of MovieLens

(b) recall (c) f-measure

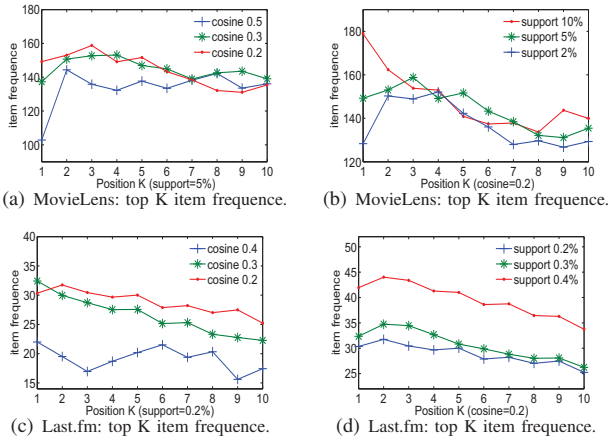
Fig. 8: CORE vs. SVD on Different Distributions

Fig. 8 (b) and (c) show the comparison in terms of recall and f-measure, respectively, on five new data. As can be seen, with the distribution of `MovieLens` getting closer to `Last.fm`,

i.e., tails getting longer, the gap on accuracy between SVD and CORE is gradually bridged. Better yet, on data 'MovieLens-40%' and 'MovieLens-50%', CORE even outperforms SVD. As we have expected, on data with more serious long tail problem, CORE is superior to classical CF methods like SVD.

Flexible Recommendation. The superiority of CORE beyond accuracy is more than challenging long tail problem. It is also flexible to tune *cosine* and *support* to control the popularity of recommended items. Fig. 9 shows the trends of the popularity of recommended items at different positions of top K under different *support* and *cosine*. A closer look at Fig. 9 (a) and (c) reveals that when *support* is constant, the higer *cosine* is, the less popular the recommended items, and as Fig. 10 (c) shows, the larger the ratios of recommended niche items. This is because when *cosine* threshold is improved, some cross-patterns (spurious relation) get filtered out. On the contrary, as Fig. 9 (b) and (d) and Fig. 10 (b) and (d) show, when *cosine* is constant, the higher *support* is, the more popular the recommended items, and the smaller the ratios of recommended niche items. This is because only items in patterns with higher *support*(more common itemsets) get to be recommended. The only exception in Fig. 10 (a) is due to the fact that *cosine* $= 0.5$ is a pretty high threshold for validating interesting patterns and thus most patterns are discarded, leading to rather limited coverage of candidate items. In real applications, we could determine appropriate *cosine* and *support* setting by referring to the amount of patterns mined.

As a result, CORE could be tuned freely to work under different circumstances, e.g. either when the system mainly wants to promote popular or niche items.



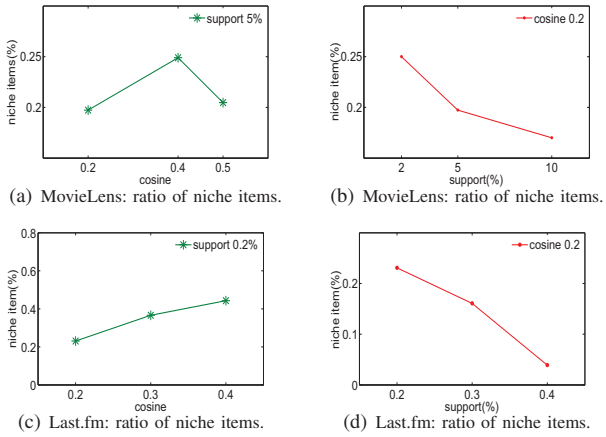
(a) MovieLens: top K item frequence. (b) MovieLens: top K item frequence.

(c) Last.fm: top K item frequence. (d) Last.fm: top K item frequence.

Fig. 9: Popularity of Recommended Items

V. Related Work

There are mainly two types of recommendation approaches, i.e., content-based and CF (collaborative filtering)-based. CF recommender systems are more popular for not demanding extra descriptive item features. Applying pattern-based techniques for recommendation is among the initial exploring of CF recommender systems. One of the earliest researches is



(a) MovieLens: ratio of niche items. (b) MovieLens: ratio of niche items.

(c) Last.fm: ratio of niche items. (d) Last.fm: ratio of niche items.

Fig. 10: Ratio of Recommended Niche Items

done by [21] which uses AR(association rules) to recommend movies. Some following work include [17], [26]. Most previous association analysis based recommenders use association rules, yet the confidence as the measure of rule interestingness may not disclose truly interesting relationships [14], [13] and the *support*-based pruning strategy is not effective for data sets with skewed *support* distribution [33]. These defects would result in either mining too many spurious patterns or a limited coverage of candidate items, leading to inaccurate recommendations.

In the last decade, many classical CF-based methods develop rapidly, of which one commonly used is neighborhood-based approach [28], [9]. Matrix factorization methods have gained popularity in recommender systems due to their effectiveness in improving accuracy [5] and have been extended to solve problems like sparsity or temporal dynamics [19]. In recent years, probabilistic topic models also become more popular [22] [7]. Most of these advanced models are inherently difficult to adapt to real-world challenges, e.g., incremental learning or distributed computing. Considering previous pattern-based methods have an edge in adapting to real-world applications, our work revisits pattern-based methods to explore better way of applying association analysis to recommender systems.

To evaluate different recommendation techniques, various metrics have been employed. Traditionally, accuracy metrics like Mean Absolute Error(MAE), Root Mean Squared Error(RMSE) and precision, recall [15] are mostly used. Yet, more and more recent studies point out that recommender systems emphasizing accuracy alone can even result in sales diversity reduction [10]. Different strategies have been proposed to challenge long tail recommendation. In [23] researchers first cluster tail items into different groups and then predict ratings within each group. Research [1] re-ranks candidate items using extra attributes, intending to push some niche items into former lists. Compare to our CORE, approaches mentioned above have limitations in either neglecting the degree of long tail property or being inflexible in adjusting the popularity of recommended items.

VI. Conclusions

In this paper, we investigate cosine pattern-based recommender systems (CORE). Specifically, we proposed how to use cosine interesting patterns to generate recommendations and discussed strategies for CORE to work efficiently. Also we explored the role of *support* and *cosine* in adjusting the popularity of recommended items. Experimental results showed that CORE has its advantage in both recommending items with competitive accuracy and promoting niche items recommendations, especially on data with longer tails.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (NSFC) (71322104, 71171007, 61103229, 71372188), by the National High Technology Research and Development Program of China (863 Program) (SS2014AA012303), by the National Information Security Research Plan of China (2013A141, 2013A143), by the National Center for International Joint Research on E-Business Information Processing (2013B01035), by the Foundation for the Author of National Excellent Doctoral Dissertation of PR China (201189), and by the Program for New Century Excellent Talents in University (NCET-11-0778).

References

[1] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24:896–911, 2012.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. *Recommender systems handbook*, pages 217–253, 2011.

[3] C. Anderson. The long tail: Why the future of business is selling less of more. In *Hachette Digital, Inc.*., 2006.

[4] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.

[5] R. M. Bell, Y. Koren, and C. Volinsky. The bellkor solution to the netflix prize. 2007.

[6] E. Brynjolfsson, Y. Hu, and D. Simester. Goodbye pareto principle,hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57:1373–1386, 2011.

[7] W. Y. Chen, J. C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. In *the 18th international conference on World wide web*, WWW '2009, pages 681–690, 2009.

[8] Kim J. K. Cho, Y. H. and S. H. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23:329–342, 2002.

[9] J. Delgado and N. Ishii. Memory-based weighted majority prediction. In *SIGIR Workshop Recomm. Syst.*, SIGIR '1999, pages 39–46, 1999.

[10] D. M. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. In *the 8th ACM conference on Electronic commerce*, EC '2007, pages 192–199, 2007.

[11] D. G. Goldstein and D. C. Goldstein. Profiting from the long tail. *Harvard Business Review*, 84:24–28, 2006.

[12] I. Guy and D. Carmel. Social recommender systems. In *Proceedings of the 20th international conference companion on World wide web*, WWW '2011, pages 283–284, 2011.

[13] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining:current status and future directions. *Data Mining and Knowledge Discovery*, 15:55–86, 2007.

[14] J. Han and M. Kamber. *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann, 2006.

[15] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM TOIS*, 22:5–53, 2004.

[16] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira. *Recommender systems handbook*. Springer, 2011.

[17] P. Kazienko. Mining indirect association rules for web recommendation. *Journal of Applied Mathematics and Computer Science*, 19:165–186, 2009.

[18] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *14th ACM SIGKDD international conference on Knowledge discovery and data mining*, SIGKDD '2008, pages 426–434, 2008.

[19] Y. Koren. Collaborative filtering with temporal dynamics. In *Communications of the ACM*, pages 89–97, 2010.

[20] R. Krestel, P. Fankhauser, and W. Nejdl. Latent dirichlet allocation for tag recommendation. In *ACM conference on Recommender systems*, RecSys '2009, pages 61–68, 2009.

[21] W. Lin, S.A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommendation systems. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.

[22] Q. Liu, E. Chen, H. Xiong, C. H. Ding, and J. Chen. Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42:218–233, 2012.

[23] Y. J. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *ACM conference on Recommender systems*, RecSys '2008, pages 11–18, 2008.

[24] K. Partridge and B. Price. Enhancing mobile recommender systems with activity inference. *User Modeling, Adaptation, and Personalization*, pages 307–318, 2009.

[25] R. Paul, S. Neophytos, S. Mitesh, B. Peter, and R. John. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings ACM conference CSCW*, CSCW '1994, pages 175–186, 1994.

[26] J. J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *ACM conference on Recommender systems*, RecSys '2007, pages 105–112, 2007.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *Proceedings ACM WebKDD Workshop*, WebKDD '2000, pages 82–90, 2000.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *the 10th international conference on World Wide Web*, WWW '2001, pages 285–295, 2001.

[29] Frankowski D. Herlocker J. Schafer, J. B. and S. Sen. Collaborative filtering recommender systems. *The adaptive web*, pages 291–324, 2007.

[30] H. Shan and A. Banerjee. Mixed-membership naive bayes models. *Data Mining and Knowledge Discovery*, 23:1–62, 2011.

[31] Cao J. Wu J. Wang Y. Wu, Z. and C. Liu. Detecting genuine communities from large-scale social networks: a pattern-based method. *The Computer Journal*, 113, 2013.

[32] J. Wu, S. Zhu, H. Liu, and G. Xia. Cosine interesting pattern discovery. *Information Sciences*, 184:176–195, 2012.

[33] H. Xiong, P. N. Tan, and V. Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery*, 13:219–242, 2006.

[34] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen. Challenging the long tail recommendation. In *Proceedings of the VLDB Endowment*, VLDB '2012, pages 896–907, 2012.