Scaling up top-K cosine similarity search

Shiwei Zhu, Junjie Wu,⁎, Hui Xiong, Guoping Xia

Information Systems Department, School of Economics and Management, Beihang University, Beijing 100191, China
Management Science and Information Systems Department, Rutgers Business School - Newark and New Brunswick, Rutgers University, Newark, NJ 07102, USA

A R T I C L E   I N F O
Article history:
Received 21 September 2009
Received in revised form 23 August 2010
Accepted 23 August 2010
Available online 8 September 2010

Keywords:
Cosine similarity
Similarity search
Diagonal traversal strategy
Max-first traversal strategy

A B S T R A C T
Recent years have witnessed an increased interest in computing cosine similarity in many application domains. Most previous studies require the specification of a minimum similarity threshold to perform the cosine similarity computation. However, it is usually difficult for users to provide an appropriate threshold in practice. Instead, in this paper, we propose to search top-K strongly correlated pairs of objects as measured by the cosine similarity. Specifically, we first identify the monotone property of an upper bound of the cosine measure and exploit a diagonal traversal strategy for developing a TOP-DATA algorithm. In addition, we observe that a diagonal traversal strategy usually leads to more I/O costs. Therefore, we develop a max-first traversal strategy and propose a TOP-MATA algorithm. A theoretical analysis shows that TOP-MATA has the advantages of saving the computations for false-positive item pairs and can significantly reduce I/O costs. Finally, experimental results demonstrate the computational efficiencies of both TOP-DATA and TOP-MATA algorithms. Also, we show that TOP-MATA is particularly scalable for large-scale data sets with a large number of items.

1. Introduction

Given a large set of items (objects) and observation data about co-occurring items, association analysis is concerned with the identification of strongly related subsets of items. Association analysis has become one of the core problems in the field of data mining and plays an important role in many application domains such as market-basket analysis [2], climate studies [30], public health [10], and bioinformatics [23,37]. For example, in the market-basket study, association analysis can find patterns which are useful for sales promotion, shelf management, and inventory management. Also, in medical informatics, association analysis can be used to find drug-drug interactions.

Although numerous scalable methods have been developed for mining frequent patterns in association analysis, the traditional support-and-confidence framework has shown its limitations in finding interesting relationships [13,14,33]. To meet this critical challenge, statistical correlation or similarity measures have been exploited for mining association patterns [32], such as lift, χ² [7], coherence, and the cosine similarity. However, most of these measures are used only for post-evaluation due to the lack of the computation-friendly property.

In this paper, we focus on developing efficient algorithms for finding top-K strongly related item pairs as measured by the cosine similarity, which has been widely used as a popular similarity measure for high-dimensional data in text mining [31,43], information retrieval [5,29], and bioinformatics [20]. Indeed, searching for top-K strongly correlated item pairs has great use in many real-world applications. A good example is to suggest alternative query formulations by finding top-K similar queries based on the similarity of the search results for those queries [28]. Another example is to recommend alternative commodities by searching for top-K commodities with similar user preferences among a large number of items.

⁎ Corresponding author.
E-mail address: wujj@buaa.edu.cn (J. Wu).

0169-023X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2010.08.004
Indeed, the cosine similarity is one of the few interestingness measures that hold symmetry, triangle inequality, null-invariance [25], and cross-support properties [39]. However, many measures, such as support and lift, do not satisfy these properties. Also, the cosine similarity is very simple and has real meaning; that is, it measures the angle difference of two vectors. This makes the cosine similarity particularly useful for measuring the proximity in high-dimensional space.

To this end, we first propose a diagonal traversal algorithm named TOP-DATA to search for the top-K strongly related pairs. TOP-DATA can utilize the monotone property of a cosine upper bound to efficiently prune item pairs without computing their exact cosine values. By using the boundary vector, we further reduce the \( O(N^2) \) space complexity for saving the checking status to \( O(N) \). We then propose a novel TOP-MATA algorithm, which employs a max-first traversal strategy for the search of top-K pairs. A theoretical analysis reveals that TOP-MATA has the abilities in reducing I/O costs and saving the computation of cosine similarity for false-positive item pairs.

Finally, we have conducted experiments on various real-world data sets. Results reveal that, compared with the brute-force method, both TOP-DATA and TOP-MATA algorithms show significant computational advantages for finding top-K item pairs. In particular, TOP-MATA has a better performance than TOP-DATA on large-scale data sets with a large number of items. We also compare TOP-MATA with TKCP[16], a well-established algorithm for finding top-K pairs using the FP-tree[15]. Results reveal that, when \( K \) is small with respect to the number of all pairs, TOP-MATA has a better performance due to the efficient pruning effect by the max-first traversal strategy.

1.1. Outline

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3, we introduce some preliminaries and define the problem. Section 4 proposes the TOP-DATA algorithm. In Section 5, we give the TOP-MATA algorithm and provide a theoretical analysis. Section 6 shows experimental results. Finally, we conclude the work in Section 7.

2. Related work

2.1. Interestingness measures

In the literature, there are many interestingness measures for mining interesting patterns from large-scale item-based data sets. For example, Piatetski–Shapiro proposed the statistical independence of rules as an interestingness measure [26]. Brin et al. [7] proposed lift and \( \chi^2 \) as correlation measures and developed an efficient mining method. Blanchard et al. [6] designed a rule interestingness measure, Directed Information Ratio, based on information theory. Hilderman and Hamilton [17] and Tan et al. [32] provided well-organized comparative studies for the interestingness measures, respectively.

Among the available interestingness measures, the cosine similarity gains particular interests. For instance, Omiecinski [25] and Lee et al. [24] revealed that cosine is null-invariant and thus is a good measure for mining interesting relationships in transaction databases. Indeed, the cosine similarity is one of the few interestingness measures that hold symmetry, triangle inequality, null-invariance [32], and cross-support properties [39]. In [34], Wu et al. extended some measures including the cosine similarity to the multi-itemset case, which further shows the value of the cosine similarity for association analysis. The work in this paper focuses on the computation of the cosine similarity for item pairs in high-dimensional space.

2.2. Mining interesting patterns

A key challenge in using the above interestingness measures is the lack of computation-friendly properties, such as the anti-monotone property [1], which can be exploited for the efficient computation. Therefore, it is hard to incorporate the measures directly into the mining algorithms. Indeed, most of the existing measures, such as cosine and \( \varphi \)-coefficient, are often used as the post-evaluation measures due to the above reason [32].

To meet this challenge, Xiong et al. introduced an upper bound to \( \varphi \)-coefficient, and proposed the TAPER algorithm to efficiently compute the all-strong-pairs correlations [38]. Along the same line, Ilyas et al. [18] also proposed a method for efficiently identifying correlated pairs, which applied the sampling techniques. Due to the nature of sampling, this method cannot avoid finding false-positive and false-negative correlations. Zhang and Feigenbaum [42] also proposed a simple randomized algorithm whose false-negative probability is negligible. Zhou and Xiong [44] proposed the CHECK-POINT algorithm to efficiently compute the all-strong-pairs correlations for stream data sets by setting a checkpoint to establish a computation buffer. These methods, however, require an interestingness threshold, which is often hard to find in practice.

To address the above problem, Xiong et al. [36,40] proposed the TOP-COP algorithm to efficiently compute the top-K correlated pairs, which is the first study to make use of the diagonal traversal strategy. Our TOP-DATA algorithm in this study also employs a similar diagonal traversal strategy for the cosine similarity, but we refine it to TOP-DATA-R by using a boundary vector. More importantly, we develop a max-first traversal strategy and propose the TOP-MATA algorithm. Theoretical and experimental studies show the advantages of TOP-MATA over TOP-DATA-R in saving the computational costs and I/O costs. He et al. [16] also proposed an algorithm called TKCP to find the top-K correlated pairs by using the FP-tree [15]. However, TKCP requires traversing all the pairs, whereas TOP-MATA avoids false-positive pairs using the max-first traversal method. As a result, when \( K \) is not very large, TOP-MATA has a significant pruning effect and shows better performances than TKCP. Details are available in the Experimental section.
The problem of finding interesting pairs has a close relationship to the all pairs similarity search problem in information retrieval community, which has been used in many application domains [21,22,27]. Early researches in this area paid close attention to using various approximation techniques [11,12,19] to solve this problem, but a recent trend is to compute the similarity and find all pairs exactly. There are three main methods which are index-based methods [5,8], prefix filtering-based methods [5,9,35] and signature-based methods [3], respectively. The above studies focus on finding binary or non-binary pairs with some specific similarity measures above some given thresholds. Recently, Awekar et al. [4] studied the problem of searching candidate pairs incrementally for varying similarity thresholds. Xiao et al. [35] studied the top-K set similarity joins problem for near duplicate detection, which enumerated all the “necessary” similarity thresholds in the decreasing order until the top-K set had been found. From this aspect, it can be also viewed as an incremental algorithm using varying similarity thresholds. In contrast, the work in this study aims to find interesting pairs with top-K cosine similarity values without setting any interestingness threshold. What’s more, TOP-MATA can avoid the computation of false-positive pairs using a novel max-first traversal strategy.

3. Preliminaries and problem definition

Cosine similarity is a measure of similarity between two high-dimensional vectors. In essence, it is the cosine value of the angle between two vectors.

\[
\cos(X, Y) = \frac{\langle X, Y \rangle}{\|X\| \|Y\|}
\]

where \(\langle \cdot \rangle\) indicates the inner product of two vectors, and \(\| \|\) indicates the L2 norm of the vector. For vectors with non-negative elements, the cosine similarity value always lies in the range of [0, 1], where 1 indicates a perfect match of two vectors, whereas 0 indicates a complete opposite.

In real-world applications, a vector often consists of binary attribute values, either 0 or 1. For instance, the purchase record of a commodity is a typical binary vector, where “1” indicates the purchase of this commodity in some transaction whereas “0” indicates the omission in another transaction. For another example, in document data sets, a word can be represented by a binary vector where each dimension indicates whether the word appeared in a particular document or not. For a pair of binary vectors, we can extract the binary information into a two-way table, as shown in Table 1. In the table, \(n_{ij}\) denotes the number of dimensions whose attribute values for X and Y are i and j, respectively. Finally, \(n+, n_0, n_{0+}, n_{+0}\) and \(N\) are defined as follows:

\[
\begin{align*}
\text{cos}(X, Y) &= \frac{n_{11}}{n_1 + n_+} \quad \text{(2)} \\
\cos(X, Y) &= \frac{\text{supp}(XY)}{\sqrt{\text{supp}(X) \text{supp}(Y)}} \quad \text{(3)}
\end{align*}
\]

From a support measure point of view adopted by the frequent pattern mining, for two items X and Y in a market-basket data set, we have \(\text{supp}(XY) = n_{11} / N\), \(\text{supp}(X) = n_1 / N\), and \(\text{supp}(Y) = n_+ / N\). Thus the cosine similarity for binary pairs can be further transformed equivalently to:

Table 1
A two-way table of X and Y.

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>(\Sigma_{\text{row}})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>(n_{00}) (n_{01}) (n_{0+})</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>(n_{10}) (n_{11}) (n_{+1})</td>
</tr>
<tr>
<td>(\Sigma_{\text{col}})</td>
<td>(n_{+0}) (n_{+1}) N</td>
<td></td>
</tr>
</tbody>
</table>
4. Finding top-K cosine similarities: the diagonal traversal algorithm

In this section, we study how to find top-K cosine similarities using a diagonal traversal strategy. We first give a simple upper bound for the cosine similarity as follows.

4.1. The upper bound of cosine similarity

For two items X and Y, without loss of generality, we assume \( \text{supp}(X) \geq \text{supp}(Y) \). According to the definition of support, we have

\[
\text{supp}(XY) \leq \text{supp}(Y).
\]

Accordingly,

\[
\cos(X, Y) = \frac{\text{supp}(XY)}{\sqrt{\text{supp}(X)\text{supp}(Y)}} \leq \frac{\text{supp}(Y)}{\sqrt{\text{supp}(X)\text{supp}(Y)}} = \sqrt{\frac{\text{supp}(Y)}{\text{supp}(X)}}
\]

(4)

Thus we get an upper bound for \( \cos(X, Y) \), which is denoted by

\[
\text{upper}(\cos(X, Y)) = \sqrt{\frac{\text{supp}(Y)}{\text{supp}(X)}}
\]

(5)

Next, we formulate the property of the upper bound.

**Property 1 Monotonicity.** Given two items X and Y with \( \text{supp}(X) \geq \text{supp}(Y) \), \( \text{upper}(\cos(X, Y)) \) is monotonously increasing (decreasing) in \( \text{supp}(Y) (\text{supp}(X)) \), given \( \text{supp}(X) (\text{supp}(Y)) \) is fixed.

**Remark.** Since the proof of Property 1 is straightforward, we omit it here. The merits of \( \text{upper}(\cos(X, Y)) \) lie in two aspects. First, the computation of the upper bound is much more efficient than that of the original cosine similarity for avoiding the computation of \( \text{supp}(X, Y) \). This is extremely important when the number of items is very large, say over 1,000,000. Second, the upper bound has a monotone property which is crucial for the efficient search of the top-K binary pairs. We detail it below.

4.2. The diagonal traversal algorithm

Here, we introduce a diagonal traversal procedure to find the binary pairs with top-K cosine similarity values (hereinafter we call them the “top-K pairs” for simplicity). The key point of this algorithm is the use of the upper bound of cosine similarity and its monotone property.

4.2.1. The filtering effect of the upper bound

Suppose we have \( n \) items. Intuitively, we can build and maintain the list of top-K pairs during the computation of the cosine similarity values for all \( n(n-1)/2 \) pairs. The updating rule for the top-K list is as follows: 1) compute the cosine value of a new binary pair; 2) if this value is larger than the minimum one of the current top-K pairs, the new pair should replace the minimum pair. However, the cost of computing directly the 2-item support \( \text{supp}(X, Y) \) is very high. So the first point is to filter item pairs with small cosine upper bounds. Let \( \text{minCos} \) denote the minimum cosine value of the pairs in the current top-K list, we have a simple lemma as follows.

**Lemma 1.** A new pair \( (X, Y) \) can enter the top-K list only if

\[
\text{upper}(\cos(X, Y)) \geq \text{minCos}.
\]

**Remark.** Since the proof of this lemma is straightforward, we omit it here. According to this lemma, we can use the upper bound as a filter to filter out the binary pairs that cannot enter the top-K list without computing their exact cosine similarity values. That is, for any new pair \( (X, Y) \), we first compute \( \text{upper}(\cos(X, Y)) \). If \( \text{upper}(\cos(X, Y)) \geq \text{minCos} \), we further compute \( \cos(X, Y) \) to check whether \( \cos(X, Y) \geq \text{minCos} \). Otherwise, we simply skip this pair without computing \( \cos(X, Y) \). We hereby call it the “filtering effect” of the upper bound. Apparently, if we can find a top-K list with a high \( \text{minCos} \) at the initial stage of the whole computational process, the filtering effect will be significant.

4.2.2. The pruning effect of the upper bound

The next task is to further reduce the cosine similarity computations by designing an efficient pruning strategy. To this end, we should make use of the monotone property of the upper bound.
4.2.2.1. The sorted item-matrix. Suppose we have \( n \) items: \( \{X_1, X_2, \ldots, X_n\} \). We first sort the items in a support-descending order. This leads to a sorted item-sequence

\[
S = \{X_1, X_2, \ldots, X_n\},
\]

where \( \text{supp}(X_i) \geq \text{supp}(X_j) \) if \( i < j \). Based on the sorted item-sequence, we can build a two-dimensional sorted item-matrix \( M = S \otimes S \), as shown in Fig. 1 \((n = 6)\). Note that since the cosine similarity is symmetric, we only need to consider the upper triangle matrix \( U_M \).

4.2.2.2. The diagonal traversal procedure. Here we illustrate the diagonal traversal procedure by the sorted item-matrix. Assume that we need to find top-2 item pairs out of the 15 pairs of 6 items as shown in Fig. 1. We employ a diagonal traversal strategy along the item-matrix. In other words, in the initial stage, we push \( P[1, 2] \) and \( P[2, 3] \) (\( P[i, j] \) is the pair of item \( i \) and item \( j \), given \( i \leq j \)) into the top-2 list, and compute their cosine values. Then, in the updating stage, we traverse along the diagonals (denoted by the dash-dotted line) in the sorted item-matrix to check in sequence whether \( P[3, 4], P[4, 5], P[5, 6], P[4, 6], P[3, 5], \ldots, P[1, 6] \) can enter the top-2 list, as shown in Fig. 1. The checking procedure for each pair consists of two phases, one for the computation of the upper bound, and the other for the computation of the exact cosine similarity if the upper bound exceeds \( \text{minCos} \), the minimum cosine value for the pairs in the current top-2 list.

4.2.2.3. The pruning effect. The diagonal traversal procedure is of great help in pruning item pairs. First, it helps to find a higher threshold, i.e., \( \text{minCos} \), for the top-\( K \) list at the beginning. Recall Eq. (5) that \( \text{upper}(\cos(X, Y)) = (\text{supp}(Y)/\text{supp}(X))^{0.5} \), given \( \text{supp}(Y) \leq \text{supp}(X) \). Since the diagonal traversal examines first the sorted items in the neighborhood, it tends to find item pairs with higher cosine upper bounds at the very beginning. As a result, these pairs may probably have higher cosine values such that \( \text{minCos} \) for the initial top-\( K \) list is high.

More importantly, the diagonal traversal procedure can make use of the monotonicity of the upper bound to further prune the item pairs. To illustrate this, we have the following theorem.

**Theorem 1.** Given the current top-\( K \) list and its \( \text{minCos} \) in a diagonal traversal procedure, if \( \text{upper}(\cos(P[k, l])) \leq \text{minCos} \), then \( \forall 1 \leq k \leq i \) and \( j \leq l \leq n \), \( \text{upper}(\cos(P[k, l])) \leq \text{minCos} \).

**Proof.** Since \( k \leq i \leq j \leq l \), we have

\[
\text{supp}(X_k) \geq \text{supp}(X_i) \geq \text{supp}(X_j) \geq \text{supp}(X_l).
\]

Therefore, by Eq. (5),

\[
\text{upper}(\cos(P[k, l])) = \sqrt{\frac{\text{supp}(X_l)}{\text{supp}(X_k)}} \leq \sqrt{\frac{\text{supp}(X_j)}{\text{supp}(X_i)}} \leq \sqrt{\frac{\text{supp}(X_i)}{\text{supp}(X_l)}} = \text{upper}(\cos(P[i, j])) \leq \text{minCos}.
\]

Therefore, the proof holds. 

![Fig. 1. The sorted item-matrix: an example.](image-url)
Remark. Theorem 1 implies that if $P_{i,j}$ cannot enter the top-$K$ list due to its small upper bound, then $P_{k,l}$ will also fail to do that given $k \leq i$ and $l \geq j$. For example, for the sorted item-matrix in Fig. 1, if $P_{3,4}$ cannot enter the top-2 list for upper($\cos(P_{3,4})$) $\leq \min\cos$, then all the pairs in the upper right corner of $P_{3,4}$ will also fail to enter the list, as shown by the shadowed area in Fig. 1.

4.2.3. TOP-DATA: the pseudocode

Here we propose an algorithm named TOP-DATA (Top-K cosine similarity Pairs using DiagonAl TrAversal method). Fig. 2 shows its pseudocode.

Line 2 initializes topKPairs to store the up-to-date top-$K$ item pairs. Since $K$ is usually not very large, the data structure for topKPairs can simply be an array of size $K$. Line 3 initializes status to store the up-to-date pruning information of each item pair. If $N$ is not very large, status can also be an array of size $N(N-1)/2$, where 0 and 1 indicate “pruned” and “not pruned” statuses, respectively.

The diagonal traversal procedure starts from Line 4 to Line 29. For each item pair on a diagonal, Line 7 checks its pruning status to determine whether to verify this pair or not. If the status is 1, Line 8 computes the cosine upper bound of this pair. If the upper bound is less than minCos, i.e., the minimum cosine value among the top-$K$ pairs, the item pair can be safely pruned, and the pruning status will be updated accordingly in Line 10. Otherwise, we will proceed to check whether the support and the real cosine values of the item pair can exceed minSupp and minCos, respectively. If so, the update procedure for the top-$K$ pairs will be called in Line 19. It is noteworthy that we employ the vertical data layout [41] to facilitate the computation of 2-item supports for cosine similarity. That is, for an item pair, we search the vertical data layout for the transaction identifier list (TID-list) of each item, and then get the 2-item supports by intersecting the TID-lists of the two items.

The TOP-DATA Algorithm

Input:
- $S$: An item list sorted by item supports in decreasing order.
- $K$: The number of item pairs requested.
- $D$: The binary data for all the items.
- $\min\supp$: The minimum support threshold.

Output:
The item pairs with top-$K$ cosine similarity values.

Variables:
- $N$: The size of $S$.
- topKPairs: The current top-$K$ item pairs.
- minPair: The item pair in topKPairs with minimum cosine value.
- minCos: The cosine similarity value of minPair.
- status: The pruning status for each item pair.

Procedure: TOP-DATA($S, K, D$)
1. $N = \text{size}(S)$, $r = 1$, $\min\cos = -1$; // $r$ - diagonal index
2. InitializeTopKPairs(topKPairs, $K$);
3. InitializePruningStatus(status, $N$);
4. while ($r < N$)
5.  i = 0, flag = 0; // $i$ - row index, flag - stopping flag
6.  for $j = r$ to $N - 1$ do // $j$ - column index
7.    if NotPruned($i, j, status$) then
8.      upper = $\sqrt{\supp[S[i]]/\supp[S[j]]}$;
9.      if ($upper < \min\cos$) then
10.     UpdatePruningStatus($i$, $j$, $status$, $N$);
11.    else
12.      flag = 1;
13.      Compute $\supp[S[i], S[j]]$ via the vertical layout of $D$;
14.      if $\supp[S[i], S[j]] < \min\supp$ then
15.        $i++$; continue;
16.      end
17.      $cos = $ $\supp[S[i], S[j]]/\sqrt{\supp[S[i]]/\supp[S[j]]}$;
18.      if $cos > \min\cos$ then
19.        UpdateTopKPairs($S[i], S[j], cos$, topKPairs, $K$);
20.    end
21. end
22. end
23. $i++$; // next row
24. end
25. if flag = 0 then // stop condition
26.  break;
27. end
28. $r++$; // next diagonal
29. end
30. return topKPairs;

Fig. 2. The TOP-DATA algorithm.
Note that we introduce a minimum support threshold: \textit{minSupp} to TOP-DATA, and use it to prune infrequent item pairs from Line 14 to Line 16. Actually there is a longstanding dilemma in finding interesting patterns. That is, a robust interesting pattern is expected to have a high support value, whereas some real interesting patterns occur rarely in the transactions, for example, the purchase of the expensive earrings and ring together. That is why we introduce \textit{minSupp} here. This indeed gives us extra flexibility. That is, we can set a high threshold to get robust patterns, or a low or even zero threshold to obtain surprising patterns. Consider that the support computation is the by-product of the cosine computation, the introduction of \textit{minSupp} does not increase the complexity of TOP-DATA.

Fig. 3 shows the details of some important sub-procedures of the TOP-DATA algorithm. Procedure \texttt{UpdatePruningStatus} updates the item pairs’ pruning statuses in the upper right corner of the sorted matrix according to Theorem 1. Procedure \texttt{UpdateTopKPairs} is called in the refinement process for the survival item pair. Two scenarios should be handled separately. Lines 1–2 are for the case that the top-$K$ list is still under construction in the initial stage, whereas Lines 3–5 are for the case that the top-$K$ list is full already.

4.3. Algorithm refinement

In this subsection, we consider the challenges raised by the large $N$ and $K$ values to the TOP-DATA algorithm. A large item number $N$ will lead to an exponential increase of the space requirement for \textit{status} whose space complexity is $O(N^2)$. At the extreme case, say $N = 10^6$, TOP-DATA may crash due to the unavailability of enough computer memory for \textit{status}. A large $K$ value, on the other hand, indicates a higher time-consumption for the maintenance of top-$K$ pairs, since we need to update \textit{minPair} and \textit{minCos} from time to time. To meet these challenges, in what follows, we refine TOP-DATA by introducing some delicate data structures. Hereinafter we call the refined TOP-DATA algorithm: TOP-DATA-R.

4.3.1. Boundary vector for the pruning status

To meet the big $N$ challenge, we use a vector of linear scale to record the pruning boundary. We call this vector the “boundary vector”.

A boundary vector $B$ is merely a one-dimensional array of size $N - 1$, which provides the up-to-date boundaries for the pruning decisions of item pairs. Initially, $N + 1$ is assigned to each entity of $B$ to guarantee that no item pair will be pruned in the beginning. As the diagonal traversal starts, $B$ should be updated according to the following two criteria:

- For any item pair $P_{i,j}$ ($1 \leq i < j \leq N$), if upper$(\cos(P_{i,j})) \leq \minCos$, let $B[i] = j$.
- For $1 \leq i, B[i] > j$, let $B[i] = j$.

To illustrate this, let us take the sorted matrix in Fig. 4 as an example. As can be seen, altogether there are 10 items, and we follow the diagonal traversal procedure to find the top-2 pairs, as shown by the red dash lines. The corresponding boundary vector is an array of size $10 - 1 = 9$, and all the entities are initiated by $10 + 1 = 11$, as shown by the vector of “stage 0” in Fig. 4.

Now, suppose we first have upper$(\cos(P_{3,4})) \leq \minCos$. According to the first criterion, $B[3] = 4$. Since $B[1] = B[2] = 11 > 4$, we also let $B[1] = B[2] = 4$, which results in the new boundary vector of “stage 1” in Fig. 4. Similarly, when we meet the next pair $P_{7,8}$ with upper$(\cos(P_{7,8})) \leq \minCos$, we have $B[7] = 8$, and thus $B[6] = B[5] = B[4] = 8$. However, since $B[3] = 4 < 8$, we will not change the values of $B[1], B[2]$ and $B[3]$. The vector of “stage 2” shows the updated values. Next, suppose $P_{4,7}$ is the third pair with upper$(\cos(P_{4,7})) \leq \minCos$, the boundary vector will be further updated to the one of “stage 3” accordingly.

Now, given the asymptotic boundary vector above, we have the following asymptotic boundary vector to decide whether an item pair should be pruned or not.

\textbf{Theorem 2.} For any item pair $P_{i,j}$, if $j \geq B[i]$, then $P_{i,j}$ cannot enter the top-$K$ list and thus can be safely pruned.

The proof is straightforward according to Theorem 1, so we omit it here. It is noteworthy that the stopping criterion for the diagonal traversal procedure using the boundary vector is exactly the same as the original TOP-DATA algorithm. That is, if all the pairs on one diagonal have cosine upper bounds no more than \textit{minCos}, then the searching of the top-$K$ list should be suspended.

\begin{verbatim}
UpdatePruningStatus (i, j, status, N)
  1. for p = 0 to i do
  2.     for q = j to N - 1 do
  3.       status[p][q] = 0;
  4.   end
  5. end

UpdateTopKPairs (X, Y, cos, topKairs, K)
  1. if (size(topKairs) < K) then
  2.     topKairs[size(topKairs)] = {X, Y, cos}; //add a pair into topKairs
  3. else
  4.     Substitute minPair in the topKairs with the new item pair {X, Y, cos}.
  5.     Traverse topKairs, select minPair and update minCos.
  6. end

Fig. 3. Some sub-procedures of the TOP-DATA algorithm.
\end{verbatim}
For example, in the above case, after the traversal of the third diagonal, since the only one not pruned item pair \(P_{[4, 7]}\) has cosine upper bound less than \(\minCos\), we can safely stop our searching and return the current top-2 list as the final result. And the final boundary vector, i.e., the one of “stage 3”, is indicated by the shaded areas of Fig. 4.

The pseudocode of the new sub-procedures using a boundary vector is shown in Fig. 5. Note that \(\text{status}\) here represents a boundary vector now. By using the boundary vector, the \(O(N^2)\) space complexity of \(\text{status}\) can be reduced greatly to \(O(N)\).

4.3.2. Min-heap for top-\(K\) pairs

In the TOP-DATA algorithm, we need to update the top-\(K\) pairs frequently during the diagonal traversal, so the time complexity is \(O(K)\) if a 1-dimensional array is used for the top-\(K\) pairs. So we use the data structure: min-heap instead. By using min-heap for the top-\(K\) pairs, the time cost for maintaining the top-\(K\) pairs can be reduced to \(O(\log K)\) in the worst. In fact, since the items have been ordered decreasingly in supports, the time-consumption is lower than \(O(\log K)\), say, close to a constant in our experiments.

5. Finding top-\(K\) cosine similarities: the max-first traversal algorithm

In this section, we revisit the top-\(K\) cosine similarity computation problem, and propose TOP-MATA (Top-\(K\) cosine similarity Pairs using MAX-first TrAversal method) for very large-scale data sets.

5.1. Why TOP-MATA?

In the previous sections, two points for TOP-DATA remain unsolved as follows:

First, although TOP-DATA has a good pruning effect, its threshold for pruning, i.e., \(\minCos\), increases gradually and is merely asymptotically optimal. Ideally, the optimal threshold should be the cosine value of the \(K\)th pair in the final top-\(K\) list, which we cannot know until we get the complete top-\(K\) pairs finally. As a result, if we use TOP-DATA, we may allow the computations of many “false-positive” item pairs; that is, we may compute the cosine values of many item pairs whose cosine upper bounds are actually smaller than the optimal threshold but larger than the current \(\minCos\). This further increases the computational costs of TOP-DATA. Therefore, as a new algorithm, \textit{TOP-MATA should have the ability to avoid the computations of cosine values for false-positive pairs.}

Second, let us take market-basket data for example, how to further lower the computational cost for 2-item supports, given that the number of transactions is very large? The diagonal traversal strategy employed by TOP-DATA indeed helps to find a higher entry threshold at the initial stage, but it also suffers from the high I/O cost that stems from the multiple I/O operations over the vertical data layout.\footnote{The data set is usually transformed into a vertical data layout for speeding up the reaches of items [41].} Therefore, as a new algorithm, \textit{TOP-MATA should have the ability to reduce the number of I/O operations over the data set.}
It is noteworthy that the item pairs whose cosine upper bounds are greater than the optimal threshold but cannot enter the final top-K list are not the false-positive pairs in our study. For these pairs, we must compute their cosine values even if we know exactly the optimal threshold at the beginning of the computations.

5.2. TOP-MATA: the algorithmic details

Similar to TOP-DATA, TOP-MATA also makes use of the filtering effect and pruning effect of the cosine upper bounds. The main difference is, TOP-MATA employs a max-first traversal strategy rather than the diagonal traversal strategy.

5.2.1. The max-first traversal procedure

The max-first traversal procedure also takes use of the sorted item-matrix introduced for TOP-DATA. Roughly speaking, it is an iterative procedure. In each iteration, it selects one row in the upper triangular matrix to check whether there is any pair that can enter the top-K list. Two points should be addressed here:

• First, the selection criterion is “max-first”; that is, the row containing the item pair with the maximum cosine upper bound will be selected.
• Second, one checking is not for one pair only, but includes all the pairs in the same row that have the same maximum upper bound value.

The whole iteration process will be suspended if the maximum upper bound is no greater than minCos, i.e., the minimum cosine value of the current top-K pairs. In what follows, we illustrate the pseudocode of TOP-MATA.

5.2.2. The pseudocode of TOP-MATA

Fig. 6 shows the pseudocode of TOP-MATA. From Line 2 to Line 5, we initialize a max-heap data structure named pairUpperHeap to store the up-to-date maximum cosine upper bound of each row in the sorted matrix. It is trivial to note that the maximum cosine upper bound of each row is exactly the one of the left-most pair remained unchecked in this row (considering the monotonicity of the upper bound). And the use of a max-heap is for the purpose of speeding up the selection of the row with the maximum upper bound, as shown in Line 6.

Lines 7–27 are for the iterative process. We first check whether the maximum upper bound maxUpper we find is larger than minCos in Line 7. If not, we stop the iterations and return the current top-K pairs as the final result in Line 28. Otherwise, we proceed to check whether the pairs with maxUpper can enter the top-K list. Lines 8–9 are to find all the pairs with the same upper bound maxUpper, from \( P_{i,j} \) to \( P_{i,i+1} \). Then we find the TID-lists of the items in these pairs for the computations of 2-item supports, and check whether these pairs can enter the top-K list by computing their support and exact cosine values from Line 10 to Line 19.

Finally, Lines 20–26 are for the update of the max-heap. That is, we should update the maximum upper bound for the row we checked above. According to the monotonicity of the upper bound, the next pair unchecked, i.e., \( P_{i,j+1} \) is the candidate. If the cosine upper bound of \( P_{i,j+1} \) is greater than minCos, then it can be added to the max-heap to update the root, i.e., the checked \( P_{i,j} \); otherwise, we only need to disable the root and will not check this row any more. Line 26 is to find the new maxUpper after the update of the max-heap.

5.2.3. An example

Here we use a simple example to illustrate the procedure of TOP-MATA. In this example, we use a simulated data set which contains 20 transactions and 8 items, as shown in Table 2(a). Table 2(b) lists the cosine similarity values of all the item pairs computed in a brute-force manner.

Suppose that we want to find top-3 item pairs, and we let minSupp = 0. First, we sort the items into a sequence \( \{a, b, c, d, e, f, g, h\} \) by their supports in the descending order, and then form the sorted item-matrix in Fig. 7. Next, we compute the cosine upper bounds for \( \{a, b\}, \{b, c\}, \ldots, \{g, h\} \), and store them into the max-heap.
The TOP-MATA Algorithm

**Input:**
- S: An item list sorted by item supports in decreasing order.
- K: The number of item pairs requested.
- D: The binary data for all the items.
- minSupp: The support minimum threshold.

**Output:**
- The item pairs with top-K cosine similarity values.

**Variables:**
- N: The size of S.
- topKPairs: the current top-K item pairs.
- minCos: the minimum cosine value in topKPair.
- pairUpperHeap: the max-heap storing item pairs' upper values.
- maxUpper: the maximum upper value in pairUpperHeap.

**TOP-MATA(S, K, D)**

1. minCos = -1, N = Size(S);
2. for i = 0 to N - 2 do // initialize pairUpperHeap
   3. j = i + 1, upper = \(\frac{\text{supp}(S[j])}{\text{supp}(S[i])}\);
   4. Add \{i, j, upper\} to pairUpperHeap;
5. end
6. maxUpper = the upper value of pairUpperHeap's root;
7. while (maxUpper ≥ minCos)
   8. i, js = the row and column indices of pairUpperHeap's root;
   9. j0 = the item index whose item support is less then js,th item;
10. for j = js to j0 - 1 do
   11. Compute supp(S[i], S[j]) via the vertical layout of D;
   12. if supp(S[i], S[j]) < minSupp then
       13. continue;
   14. end
   15. cos = supp(S[i], S[j]) / (\text{supp}(S[i]) \cdot \text{supp}(S[j]));
   16. if cos > minCos then
       17. UpdateTopKPairs(S[i], S[j], cos, topKPairs, K);
   18. end
   19. end
20. upper = \(\frac{\text{supp}(S[j])}{\text{supp}(S[i])}\);
21. if upper > minCos then
    22. Substitute the root of pairUpperHeap with \{i, js, upper\};
    23. else
24. Substitute the root of pairUpperHeap with virtual \{i, N, -1\};
25. end
26. Heapify the max-heap of pairUpperHeap, and update the maxUpper;
27. end
28. return topKPairs;

*Fig. 6.* The TOP-MATA algorithm.

For the iteration process, we

1. Find \((g, h)\) in the max-heap with the highest upper bound 1.000. Add \((g, h)\) to the top-K list with cos = 0, and disable the node for row g in the max-heap.
2. Find \((d, e)\) in the max-heap with the highest upper bound 0.894. Add \((d, e)\) to the top-K list with cos = 0.894, and update the upper bound for row d with 0.571.
3. Find \((e, f)\) in the max-heap with the highest upper bound 0.866. Add \((e, f)\) to the top-K list with cos = 0, and update the upper bound for row e with 0.707. Set minCos = 0.
4. Find \((c, d)\) in the max-heap with the highest upper bound 0.846. Replace \((g, h)\) by \((c, d)\) in the top-K list with cos = 0.169, and update the upper bound for row c with 0.756. Set minCos = 0.
5. Find \((b, c)\) in the max-heap with the highest upper bound 0.837. Replace \((e, f)\) by \((b, c)\) in the top-K list with cos = 0.837, and update the upper bound for row b with 0.707. Set minCos = 0.169.
6. Find \((f, g)\) and \((f, h)\) in the max-heap with the highest upper bound 0.817. Replace \((c, d)\) by \((f, h)\) in the top-K list with cos = 0.817, and disable the node for row f in the max-heap. Set minCos = 0.817.
7. Terminate the iteration process for the maximal upper bound in the max-heap is 0.756 < minCos = 0.817. \((d, e), (b, c)\) and \((f, h)\) are returned as the top-3 pairs.

*Fig. 7* illustrates the TOP-MATA traversal procedure. Compared with the brute-force method, TOP-MATA saves the computations for the additional 21 cosine similarity values.
5.3. Properties of TOP-MATA

Here, we further explore the characteristics of TOP-MATA. Indeed, compared with TOP-DATA, TOP-MATA has two notable merits as follows.

5.3.1. Avoiding the false-positive pair computations

First, TOP-MATA can avoid the computations of cosine values for false-positive pairs—the pairs that have upper bounds greater than the current minCos but smaller than or equal to the optimal threshold, i.e., the minimum cosine value of the final top-K pairs which we do not know during the computational process. Let $\theta$ denote the optimal threshold, we have the following theorem.

**Theorem 3.** In the max-first procedure of TOP-MATA, given the item pair $P$ to be checked and minCos of the current top-K list, we have

$$\text{upper}(\cos(P)) > \text{minCos} \iff \text{upper}(\cos(P)) > \theta.$$ (8)

### Table 2

The example.

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a,b,c,g</td>
<td>11</td>
<td>a,d,e</td>
</tr>
<tr>
<td>2</td>
<td>a,b,c,d,e</td>
<td>12</td>
<td>a,d,e</td>
</tr>
<tr>
<td>3</td>
<td>a,b,c</td>
<td>13</td>
<td>a,f,h</td>
</tr>
<tr>
<td>4</td>
<td>a,b,c</td>
<td>14</td>
<td>a,f,h</td>
</tr>
<tr>
<td>5</td>
<td>a,b,c</td>
<td>15</td>
<td>a,f,h</td>
</tr>
<tr>
<td>6</td>
<td>a,b</td>
<td>16</td>
<td>a,g</td>
</tr>
<tr>
<td>7</td>
<td>a,b</td>
<td>17</td>
<td>a</td>
</tr>
<tr>
<td>8</td>
<td>a,b</td>
<td>18</td>
<td>a</td>
</tr>
<tr>
<td>9</td>
<td>a,d,e</td>
<td>19</td>
<td>b,c</td>
</tr>
<tr>
<td>10</td>
<td>a,d,e</td>
<td>20</td>
<td>b,c</td>
</tr>
</tbody>
</table>

### (a) The transaction data set

<table>
<thead>
<tr>
<th>Pair</th>
<th>Cosine</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,b</td>
<td>0.596</td>
</tr>
<tr>
<td>a,c</td>
<td>0.445</td>
</tr>
<tr>
<td>a,d</td>
<td>0.527</td>
</tr>
<tr>
<td>a,e</td>
<td>0.471</td>
</tr>
<tr>
<td>a,f</td>
<td>0.408</td>
</tr>
<tr>
<td>a,g</td>
<td>0.333</td>
</tr>
<tr>
<td>a,h</td>
<td>0.333</td>
</tr>
<tr>
<td>b,c</td>
<td>0.837</td>
</tr>
</tbody>
</table>

### (b) The cosine values

<table>
<thead>
<tr>
<th>Pair</th>
<th>Cosine</th>
</tr>
</thead>
<tbody>
<tr>
<td>b,d</td>
<td>0.141</td>
</tr>
<tr>
<td>b,e</td>
<td>0.158</td>
</tr>
<tr>
<td>b,g</td>
<td>0.224</td>
</tr>
<tr>
<td>c,d</td>
<td>0.169</td>
</tr>
<tr>
<td>c,e</td>
<td>0.189</td>
</tr>
<tr>
<td>c,g</td>
<td>0.267</td>
</tr>
<tr>
<td>d,e</td>
<td>0.894</td>
</tr>
<tr>
<td>f,h</td>
<td>0.817</td>
</tr>
</tbody>
</table>

Note: Pairs omitted are with cosine value 0.

Fig. 7. Illustration of the max-first procedure of TOP-MATA.
Proof. It is trivial to note that

$$\theta \geq \min \cos.$$  \hfill (9)

So the necessary condition holds. It remains to prove the sufficient condition. We use the proof by contradiction. If \( \text{upper}(\cos(P)) \leq \theta \), we have

$$\min \cos < \text{upper}(\cos(P)) \leq \theta.$$  \hfill (10)

Since the \( \min \cos \) is smaller than \( \theta \), there must be at least one unchecked item pair that should enter the final top-K list. However, according to the max-first rule of TOP-MATA,

$$\text{upper}(\cos(P)) = \max_p \text{upper}(\cos(p)) \leq \theta,$$  \hfill (11)

where \( p \) is any item pair in the sorted matrix that remains unchecked yet. This implies that there will be no more item pair that can enter the top-K list due to the filtering effect of the upper bound. This contradicts the above result. Thus, the proof holds. \( \square \)

Remark. Theorem 3 implies that although we cannot know \( \theta \) before we get the final top-K list, the max-first strategy can still guide us out of the “morass” of false-positive pairs, as if we know \( \theta \) already. This can further save the computational time of TOP-MATA.

In what follows, we compute the scale of the computation savings. To this end, we should first assume that we know the optimal threshold \( \theta \) already. Then according to the different distributions of the rank-support function, we have two cases as follows:

Case 1. The linear distribution

In this case, the rank-support function has a linear distribution: \( f(k) = a - mk \), where \( m \) and \( a \) are two parameters, as shown in Fig. 8(a). We first have the following proposition:

Proposition 1. Given the linear rank-support function \( f(k) = a - mk \) and the optimal threshold \( \theta \), the traversal boundary of TOP-MATA is

$$j = \theta^2 mi + \frac{a(1-\theta^2)}{m},$$  \hfill (12)

where \( i \) and \( j \) are rank indices for item pair \( P[i,j] \).

(a) The Rank-Support Function. (b) The Traversal Boundaries.

Fig. 8. The linear distribution case.
Proof. As we know, the cosine upper bound for an item pair \( P_{[i,j]} \) is

\[
upper\left(\cos\left(P_{[i,j]}\right)\right) = \sqrt{\frac{supp(X_i)}{supp(X_j)}} = \sqrt{\frac{a-m_j}{a-m_i}}
\]

Accordingly, given any item \( X_{[i,j]} \), TOP-MATA reaches the traversal boundary when the cosine upper bound is equal to \( \theta \). That is,

\[
\sqrt{\frac{a-m_j}{a-m_i}} = \theta \iff j = \theta^2 mi + \frac{a(1-\theta^2)}{m}.
\]

Thus, the proof holds.

The bold dash line in Fig. 8 represents the traversal boundary of TOP-MATA. We then have another proposition as follows:

**Proposition 2.** Given the linear rank-support function \( f(k) = a - mk \) and the optimal threshold \( \theta \), the traversal boundary of TOP-DATA is

\[ j = i + \frac{a(1-\theta^2)}{m}, \]  \hspace{1cm} (13)

where \( i \) and \( j \) are rank indices for item pair \( P_{[i,j]} \).

**Proof.** Since TOP-DATA takes the diagonal traversal procedure, the boundary of TOP-DATA must be \( j = i + c \). Now it remains to show \( c' = a(1-\theta^2)/m \). For simplicity, let \( c \) denote \( a(1-\theta^2)/m \).

First, assume that \( c'<c \). This implies \( upper(\cos(P_{[0,c]})) < \min Cos \) in TOP-DATA. Since \( \min Cos \leq \theta \), we have \( upper(\cos(P_{[0,c]})) < \theta \). Note that \( c \) is also the intercept of the boundary of TOP-MATA, so we have \( upper(\cos(P_{[0,c]})) = \theta \) in TOP-MATA, which results in a contradiction.

Second, assume that \( c>c \). In TOP-MATA, we have \( upper(\cos(P_{[0,c]})) = \theta \). Since the item pair whose cosine value is \( \theta \) is on the left side of the boundary of TOP-MATA, we have \( upper(\cos(P_{[0,c]})) = \min Cos = \theta \) in TOP-DATA. This implies that \( upper(\cos(P_{[0,c]})) = upper(\cos(P_{[0,c]})) \), or equivalently, \( c' = c \), which results in a contradiction. Thus, the proof holds.

The bold solid line in Fig. 8 represents the traversal boundary of TOP-DATA. Note that in Fig. 8, the slope of TOP-MATA’s boundary is always no greater than the slope of TOP-DATA’s boundary, i.e., \( \theta^2 m \leq 1 \), since we have \( m \leq 1 \) and \( \theta \leq 1 \). Accordingly, the shaded area in Fig. 8 indicates the computation saving of TOP-MATA by avoiding the false-positive pairs of TOP-MATA. We then have the following corollary.

**Corollary 1.** Given the linear rank-support function \( f(k) = a - mk \) and the optimal threshold \( \theta \), the computation saving of TOP-MATA by avoiding the false-positive pairs of TOP-DATA is

\[
\hat{\delta}_c = \frac{1}{2} \left( N - \frac{a(1-\theta^2)}{m} \right) \left( \frac{1}{m\theta^2} - 1 \right),
\]

where \( N \) is the number of items.

**Proof.** The shaded area in Fig. 8 is a triangle. It is easy to show by the two boundary functions that the width and height of the triangle are

\[
W = \frac{N-c}{m\theta^2} - (N-c), \quad \text{and} \quad H = N-c,
\]

respectively. Therefore, \( \hat{\delta}_c = WH/2 \). Thus, the proof holds.

**Remark.** Four parameters, namely \( m, a, \theta \) and \( N \), determine the scale of the computation saving of TOP-MATA. It is obvious that \( N \) is a positive parameter to \( \hat{\delta}_c \); that is, given other parameters unchanged, the larger the scale of the items in the data set, the more computation savings brought by TOP-MATA. In other words, compared with TOP-DATA, TOP-MATA is more suitable for large-scale data sets with multiple items.

**Case 2. The generalized Zipf distribution.**

In this case, the rank-support function has a generalized Zipf distribution: \( f(k) = \frac{c}{k^p} \), where \( c \) and \( p \) are constants and \( p \geq 1 \), as shown in Fig. 9(a). When \( p = 1 \), \( f \) reduces to a Zipf distribution. We first have the following propositions.
Proposition 3. Given the rank-support function \( f(k) = \frac{c}{k^p} \) and the optimal threshold \( \theta \), the traversal boundary of TOP-MATA is

\[
j = \theta - \frac{2}{p} i,
\]

where \( i \) and \( j \) are rank indices for item pair \( P[i,j] \).

Proposition 4. Given the rank-support function \( f(k) = \frac{c}{k^p} \) and the optimal threshold \( \theta \), the traversal boundary of TOP-DATA is

\[
j = i + N \left( 1 - \theta^\frac{2}{p} \right),
\]

where \( i \) and \( j \) are rank indices for item pair \( P[i,j] \).

The proofs for Propositions 3 and 4 are similar to Propositions 1 and 2, respectively. So we omit the proofs here. The bold dashed and solid lines in Fig. 9 represent the traversal boundaries for TOP-MATA and TOP-DATA, respectively. Note that in Fig. 9, the slope of TOP-MATA’s boundary is always no less than the slope of TOP-DATA’s boundary, i.e., \( \theta - \frac{2}{p} \geq 1 \), since we have \( 1/\theta \geq 1 \) and \( 2/p > 0 \). Accordingly, the shaded area in Fig. 9 indicates the computation saving of TOP-MATA by avoiding the false-positive pairs of TOP-MATA. We then have the following corollary.

Corollary 2. Given the rank-support function \( f(k) = \frac{c}{k^p} \) and the optimal threshold \( \theta \), the computation saving of TOP-MATA by avoiding the false-positive pairs of TOP-DATA is

\[
\delta_c = \frac{1}{2} N^2 \left( 1 - \theta^\frac{2}{p} \right) \theta^\frac{2}{p},
\]

where \( N \) is the number of items.

Remark. The proof of Corollary 2 is very similar to Corollary 1, so we omit it here. Three parameters, namely \( N, p \) and \( \theta \), determine the scale of the computation saving of TOP-MATA. Again we can find that, given other parameters unchanged, the larger the scale of the items in the data set, the more computation savings brought by TOP-MATA. So TOP-MATA indeed has merits on handling large-scale data sets with multiple items. However, \( \theta \) and \( p \) have no such simple monotonicity. If we let \( \theta^\frac{2}{p} = 0.5 \), we have the maximum computation saving: \( \frac{1}{8} N^2 \).

In summary, TOP-MATA shows merits on saving the unnecessary computations of false-positive pairs introduced by TOP-DATA. This computation saving can be more significant for large-scale data sets with large numbers of items.

5.3.2. Saving the I/O costs

Furthermore, TOP-MATA can reduce the I/O costs for the computations of 2-item supports, which is crucial for the computational efficiency.
As mentioned above, in each iteration, TOP-MATA finds all the adjacent pairs in one row with a same cosine upper bound, then searches the vertical data layout for the transaction identifier list (TID-list) of each item, and finally gets the 2-item supports by intersecting the TID-lists of the two items (here we define the number of searches for the items’ TID-lists in the vertical data layout as the I/O cost). Since these pairs have an item in common, we only need to read the common item once in each iteration, which can save computational time considerably. Hereby we call this the “sharing effect” of TOP-MATA. Furthermore, if the data set is too large to be loaded into the memory, the computational savings by reducing the I/O costs will be more substantial.

Now, one question remains: for a real-world data set, can we estimate the level of the sharing effect of TOP-MATA before using it? To answer this question, we introduce a new index: Estimated Averaged Sharing Length, denoted by \( \text{EASL} \). In general, \( \text{EASL} \) measures the averaged number of item pairs handled simultaneously by TOP-MATA in each iteration. These pairs share a common subsequence. Let \( \text{supp}(X_j) \) denote the volume of \( X_j \), i.e., \( n_i = ||X_i|| \), \( 1 \leq i \leq t \). Moreover, we assume that TOP-MATA traverses all the pairs in the sorted item-sequence. Then we have the following proposition.

**Proposition 5.**

\[
\text{EASL} = \frac{n(n-1)}{2(n-t + nt - \sum_{i=1}^{t} n_i)}. \tag{16}
\]

**Proof.** We first count the total number of iterations TOP-MATA takes in the max-first procedure. Given any item \( X_j \in S_i \), the traversal of all the pairs in \( \cup_{1 \leq j \leq \min(k, n)} n_i \) needs only 1 iteration, and the traversal of all the pairs in \( \cup_{1 \leq j \leq \min(k, n)} \) needs \( t - i \) iterations. Therefore, altogether the traversal of all the pairs in \( P_i = \{(X_j, X_k)\} \cup_{1 \leq j \leq \min(k, n)} \) needs \( f_i = n_i - 1 + n_i(t - i) \) iterations. Since the set of all the pairs \( P = \cup_{i=1}^{t} P_i \), the total number of iterations is

\[
f = \sum_{i=1}^{t} f_i = n - t + nt + \sum_{i=1}^{t} n_i.
\]

Furthermore, since \( P = \frac{n(n-1)}{2} \), we finally have

\[
\text{EASL} = \frac{||P||}{f} = \frac{n(n-1)}{2(n-t + nt - \sum_{i=1}^{t} n_i)}. \]

Therefore, the proof holds.

**Remark.** Note that the computation of \( \text{EASL} \) is based on the assumption that TOP-MATA traverses all the item pairs. For real-world data sets, however, TOP-MATA will abort iterations when the next maximal cosine upper bound is not greater than the current threshold, as indicated by Line 7 of Fig. 6. So \( \text{EASL} \) is merely an approximation of the real averaged sharing length. Nevertheless, \( \text{EASL} \) is still valuable in that (1) the approximation is fairly accurate in real-world applications, and (2) it helps us to know the scale.

![Fig. 10. The curve of the I/O saving of TOP-MATA.](image-url)
of the sharing effect when we only get the 1-item support information. The first point is demonstrated by the experimental results in Table 8. We would like to discuss the second point in the following text.

Suppose we know the $EASL$ value now. For TOP-MATA, in each iteration, it needs to read $EASL + 1$ TID-lists for $EASL + 1$ different items in the vertical data layout. For TOP-DATA, however, it needs to reach $2EASL$ TID-lists for $2EASL$ different items. So the ratio of I/O saving of TOP-MATA is

$$\delta_{IO} = 1 - \frac{1 + EASL}{2EASL} = \frac{1}{2} \left(1 - \frac{1}{EASL}\right).$$  \hspace{1cm} (17)

According to Eq. (17), the curve of I/O saving ratio has an asymptotic line: $f(x) = 0.5$, as shown in Fig. 10. That is to say, a small $EASL$ value, say 10, can result in a big I/O saving of 45% already. Therefore, we can conclude that: (1) the I/O saving of TOP-MATA is pervasive for real-world data sets; (2) the I/O saving however has a rigorous upper bound 50%.

One major concern for TOP-MATA may be the cost of finding the maximum upper bound in each iteration. Actually this is why we introduce the max-heap to store the maximum upper bounds for each row. First, since the search operation is for the root node, so its cost is simply $O(1)$. Accordingly the major cost is for the heapification operation of the max-heap. However, according to the property of max-heap, the complexity of the heapification operation is only $O(\log n)$ in the worst, where $n$ is the total number of items. Therefore, in general, the use of max-heap for the max-first procedure guarantees the high efficiency of TOP-MATA.

6. Experimental results

In this section, we study the performances of TOP-DATA and TOP-MATA algorithms on various real-world data sets. Specifically, we aim to demonstrate: (1) the effectiveness of the diagonal traversal strategy of TOP-DATA, and (2) the merits of the max-first traversal strategy of TOP-MATA.

6.1. The experimental setup

Table 3 shows the characteristics of seven real-world data sets used in the experiments.

These data sets are from various application domains. Connect is from the UCI Machine Learning Repository\(^2\), which contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. Accidents is from the National Institute of Statistics (NIS)\(^3\) and contains (anonymized) traffic accident data for the region of Flanders (Belgium) for the period 1991–2000. Pumsb corresponds to binarized versions of a census data set from IBM.\(^4\) Re1 is from Reuters-21578 text categorization test collection Distribution 1.0\(^5\)which contains documents that appeared on Reuters newswire in 1987. Wap is from the WebACE project. Each document corresponds to a web page listed in the subject hierarchy of Yahoo!. Retail contains the (anonymized) retail market-basket data from an anonymous Belgian retail store.\(^6\) Finally, La12 was obtained

---

3 Available at http://fimi.cs.helsinki.fi/data/.
4 Available at http://fimi.cs.helsinki.fi/data/.
5 Available at http://www.daviddlewis.com/.
6 Available at http://fimi.cs.helsinki.fi/data/.
from articles of the Los Angeles Times that was used in TREC-5. The categories correspond to the desk of the paper that each article appeared and include documents from the entertainment, financial, foreign, metro, national, and sports desks.

Note that: (1) We let minSupp = 0 for both TOP-DATA and TOP-MATA for finding more interesting item pairs; (2) the original La12 data set has too many items whose support counts are only one. To avoid the noise disturbance, we have filtered out these items and constructed the refined La12 data set in Table 3; (3) since we would like to explore the algorithm performances along a series of K values for top-K pairs, we set different K series for the seven data sets according to their intrinsic data characteristics, as shown in Table 4; (4) for simplicity, hereinafter we denote the brute-force method by BF, the TOP-DATA algorithm by TD, the TOP-DATA-R algorithm by TDR, and the TOP-MATA algorithm by TM.

All the algorithms were implemented in C#, and run in the environment of Microsoft.Net 3.5, on a Microsoft Vista Ultimate platform with SP2 32bit edition. The experimental PC is with an AMD Athlon64 3800+ CPU, 4 GB DDRII 667 MHz RAM, and a Seagate Barracuda 7200.11 500GB hard disk. Table 5 shows the execution time for BF, TD, TDR, and TM, respectively.

### 6.2. TOP-DATA versus the brute-force method

First, we would like to compare the performance of the TOP-DATA algorithm with the performance of the brute-force method. Fig. 11 shows the results of data sets Connect, Pumsb, Retail and La12 (for all data sets please refer to Table 5). As can be seen, due to the filtering effect of the upper bound and the pruning effect of the diagonal traversal procedure, TOP-DATA consumes much less execution time than the brute-force method. This is particularly true for small K values. As K increases, the gap becomes smaller. Another interesting observation is that the growth of the execution time for Connect, Retail and La12 are much faster than Pumsb. This actually implies that there is more room for TOP-MATA to improve the performances of TOP-DATA on the former three data sets. We will show this in the following subsections.

### 6.3. TOP-DATA-R versus TOP-DATA

Here we illustrate the effectiveness of the boundary vector in TOP-DATA-R. Fig. 12 shows the comparison results on Connect, Rel, Retail and La12 data sets. As can be seen, the execution time of all data sets has been reduced significantly by TOP-DATA-R except for the Connect data set. The reason for this observation is that the Connect data set contains only 129 items whereas the other three data sets all contain over 3000 items, as shown in Table 3. This implies that TOP-DATA-R is particularly useful for data

---

7 Available at http://trec.nist.gov.
sets with a large number of items. Since TOP-DATA-R is generally better than TOP-DATA, we will only compare TOP-DATA-R with TOP-MATA in the following context.

6.4. TOP-MATA versus TOP-DATA-R

As mentioned above, TOP-MATA employs a max-first rather than the diagonal traversal procedure. So in this section, we would like to study the effectiveness of the max-first traversal strategy for real-world data sets. Fig. 13 shows the results of some data sets computed by TOP-MATA and TOP-DATA-R, respectively. As can be seen, in general, TOP-MATA shows a better performance than TOP-DATA-R, especially for data sets Connect, Rel, Wap, La12 and Retail. And as the increase of the \( K \) value, the advantage tends to be even more impressive for these five data sets.

Here comes the question, why TOP-MATA shows varied improvements for different data sets? As pointed out in Section 5.2.3, TOP-MATA has the merits of (1) avoiding the false-positive pair computations (“computation saving” for short) by employing the max-first strategy and (2) saving the I/O costs (“I/O saving” for short) by horizontal searches for pairs with a same upper bound. So in what follows we explore the influences of these two factors to our question.

6.5. The computation savings of TOP-MATA

In Section 5.2.3, we formulate how to compute the computation savings of TOP-MATA with the assumption that the optimal threshold \( \theta \) is already known. This is not true for real-world cases. Nevertheless, we can still get the real computation savings by comparing the number of item pairs computed\(^8\) by TOP-MATA and TOP-DATA-R in the experiments, respectively.

Table 6 shows the results. The number of items computed by TOP-MATA are recorded in the upper sub-table, and the sub-table below shows the ratios of the reduced computations to the computations of TOP-DATA-R. As can be seen in the table, four data

\(^8\) “computed” here means we need to compute the item pair’s cosine value.
sets, namely Connect, Re1, Wap and La12, enjoy significant computation savings brought by TOP-MATA. Fig. 14 also shows the computation savings for Wap and La12 directly. As to the rest three data sets, i.e., Accidents, Pumsb and Retail, the computation savings are merely marginal, typically with saving ratios not over 10%. Recall that TOP-MATA shows significant improvements on Connect, Re1, Wap and La12 in the previous section, we can conclude that the computation saving is a major factor for the performance of TOP-MATA. That is, compared with TOP-DATA-R, a higher computation saving implies a much better performance of TOP-MATA.

One problem remains: since its computation saving is relatively small in Table 6, why data set Retail still shows a notable improvement by using TOP-MATA instead of TOP-DATA-R in Fig. 13(e)? The I/O saving factor below will illustrate this nicely.

6.6. The I/O savings of TOP-MATA

Here, we evaluate the I/O savings of TOP-MATA, which refers to the savings due to the horizontal search of pairs with the same upper bound in TOP-MATA. Table 7 shows the I/O saving ratio of TOP-MATA to TOP-DATA-R. As can be seen in Table 7, the I/O savings of Connect, Accidents and Pumsb are the smaller ones among all seven data sets. And the other four data sets all have I/O savings around 50%, including Retail. Now we can answer why TOP-MATA has much better performances on Retail rather than Accidents and Pumsb (refer to Fig. 13), although these three data sets all have smaller computation savings (refer to Table 6). The main reason is, Retail has much larger I/O savings than Accidents and Pumsb. Since Accidents and Pumsb have both smaller computation savings and smaller I/O savings, TOP-MATA shows only comparable results to TOP-DATA-R, as shown in Fig. 13.

As pointed out in Section 5.2.3, we can evaluate the I/O savings of TOP-MATA by the Estimated Averaged Sharing Length (EASL). Table 8 shows the results. In the table, all the numbers are the real ASLs recorded by TOP-MATA except for the bottom line which contains the EASL values. By comparing ASL and EASL values, we know that EASL provides a good approximation to the real ASL. As can be seen, when EASL = 7.66 for Pumsb, the real I/O savings have reached about 40%, which implies that the I/O savings of TOP-MATA are indeed pervasive for real-world data sets. However, we should also notice that, the I/O savings have a ceiling of 50% even for a very high EASL value 294.15, as shown in Table 7. That means, compared with the computation savings, the I/O savings are a secondary factor for the performances of TOP-MATA.

Fig. 12. A performance comparison: TDR versus TD.
Fig. 13. A performance comparison between TOP-MATA and TOP-DATA-R.
Furthermore, it is noteworthy that TOP-MATA is more suitable for extremely large data sets which cannot be loaded into memory completely. In this case, the I/O saving effect of TOP-MATA will be more considerable. To verify this, we can simply do simulations by reading items from the hard disk rather than memory. Fig. 15 shows the performance comparison of TOP-MATA and TOP-DATA-R for Pumsb, Re1 and Wap. As can be seen, the performance improvements by TOP-MATA become more significant than the ones in Fig. 13.

In summary, TOP-MATA is superior to TOP-DATA-R according to both computational and I/O savings. Since these two savings are more significant as the increase of the items, TOP-MATA works better for large-scale data sets with a large number of items.

6.7. TOP-MATA versus TKCP

In this subsection, we provide the experiments to compare the performances between TOP-MATA and TKCP [16].

As shown in Fig. 16, when $K \leq 550$, TOP-MATA is much more efficient than TKCP on five data sets Connect, Accidents, Pumsb, La12 and Wap. This indicates that, when $K$ is relatively small, TOP-MATA has significant pruning effects on false-positive pairs. As $K$ increases, however, TOP-MATA has to traverse more pairs, and thus TKCP begins to gain advantages due to the use of FP-tree, as shown by the results on Re1 and Retail data sets. Note that, in real-world applications, the desired $K$ is often small compared with the number of all the pairs.

Finally, since we do not get the source codes from the authors of TKCP, we implemented TKCP according to the pseudocode in [16]. Both TKCP and TOP-MATA were implemented using C#.

7. Concluding remarks

In this paper, we studied the problem of searching for top-$K$ item pairs with the highest $K$ cosine values among all the item pairs. Specifically, we provided two algorithms, TOP-DATA and TOP-MATA, for efficiently performing top-$K$ cosine similarity

![Table 6](image)

The computation savings by TOP-MATA.

<table>
<thead>
<tr>
<th>$K$</th>
<th>Connect</th>
<th>Accidents</th>
<th>K</th>
<th>Pumsb</th>
<th>Re1</th>
<th>Retail</th>
<th>K</th>
<th>Wap</th>
<th>La12</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>162</td>
<td>6732</td>
<td>100</td>
<td>30,835</td>
<td>427,468</td>
<td>5,364,874</td>
<td>200</td>
<td>2,369,256</td>
<td>14,821,417</td>
</tr>
<tr>
<td>150</td>
<td>388</td>
<td>9895</td>
<td>400</td>
<td>84,605</td>
<td>2,428,183</td>
<td>33,162,052</td>
<td>400</td>
<td>2,369,256</td>
<td>14,821,417</td>
</tr>
<tr>
<td>250</td>
<td>597</td>
<td>14,182</td>
<td>700</td>
<td>169,779</td>
<td>3,096,340</td>
<td>45,177,391</td>
<td>800</td>
<td>5,148,996</td>
<td>31,628,332</td>
</tr>
<tr>
<td>350</td>
<td>857</td>
<td>16,095</td>
<td>1100</td>
<td>248,565</td>
<td>3,692,374</td>
<td>56,010,870</td>
<td>1600</td>
<td>9,337,318</td>
<td>48,234,190</td>
</tr>
<tr>
<td>450</td>
<td>1145</td>
<td>20,063</td>
<td>1400</td>
<td>303,902</td>
<td>3,985,479</td>
<td>59,388,405</td>
<td>3200</td>
<td>14,488,394</td>
<td>81,551,896</td>
</tr>
<tr>
<td>550</td>
<td>1425</td>
<td>23,335</td>
<td>1700</td>
<td>369,560</td>
<td>4,351,348</td>
<td>63,460,665</td>
<td>6400</td>
<td>18,240,140</td>
<td>110,911,860</td>
</tr>
</tbody>
</table>

The ratio of the reduced computations to the computations of TOP-DATA-R.

<table>
<thead>
<tr>
<th>$K$</th>
<th>Connect</th>
<th>Accidents</th>
<th>K</th>
<th>Pumsb</th>
<th>Re1</th>
<th>Retail</th>
<th>K</th>
<th>Wap</th>
<th>La12</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>58.35%</td>
<td>4.16%</td>
<td>100</td>
<td>18.23%</td>
<td>46.36%</td>
<td>0.01%</td>
<td>200</td>
<td>0.83%</td>
<td>23.04%</td>
</tr>
<tr>
<td>150</td>
<td>55.91%</td>
<td>3.35%</td>
<td>400</td>
<td>2.10%</td>
<td>12.12%</td>
<td>2.65%</td>
<td>400</td>
<td>30.12%</td>
<td>45.38%</td>
</tr>
<tr>
<td>250</td>
<td>53.61%</td>
<td>2.62%</td>
<td>700</td>
<td>0.53%</td>
<td>8.45%</td>
<td>1.84%</td>
<td>800</td>
<td>20.03%</td>
<td>25.95%</td>
</tr>
<tr>
<td>350</td>
<td>48.28%</td>
<td>2.37%</td>
<td>1100</td>
<td>0.35%</td>
<td>2.50%</td>
<td>3.95%</td>
<td>1600</td>
<td>13.06%</td>
<td>27.89%</td>
</tr>
<tr>
<td>450</td>
<td>43.12%</td>
<td>1.37%</td>
<td>1400</td>
<td>0.40%</td>
<td>5.09%</td>
<td>1.48%</td>
<td>3200</td>
<td>6.14%</td>
<td>12.70%</td>
</tr>
<tr>
<td>550</td>
<td>39.36%</td>
<td>0.60%</td>
<td>1700</td>
<td>0.20%</td>
<td>2.03%</td>
<td>4.00%</td>
<td>6400</td>
<td>5.63%</td>
<td>7.11%</td>
</tr>
</tbody>
</table>

![Fig. 14](image)

A comparison of computational performances between TOP-DATA-R and TOP-MATA.
search. TOP-DATA exploits a diagonal traversal strategy, while a max-first traversal strategy is employed for TOP-MATA. Both theoretical analysis and experimental studies show the effectiveness of these two algorithms. In particular, the results show that TOP-MATA is superior for handling large-scale data sets with a large number of items.

<table>
<thead>
<tr>
<th>K</th>
<th>Connect (%)</th>
<th>Accidents (%)</th>
<th>K (%)</th>
<th>Pumsb (%)</th>
<th>Re1 (%)</th>
<th>Retail (%)</th>
<th>K (%)</th>
<th>Wap (%)</th>
<th>La12 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>8.02</td>
<td>32.86</td>
<td>100</td>
<td>46.30</td>
<td>49.58</td>
<td>49.83</td>
<td>200</td>
<td>49.85</td>
<td>49.93</td>
</tr>
<tr>
<td>150</td>
<td>7.22</td>
<td>25.85</td>
<td>400</td>
<td>40.32</td>
<td>49.27</td>
<td>49.83</td>
<td>400</td>
<td>49.51</td>
<td>49.93</td>
</tr>
<tr>
<td>250</td>
<td>6.53</td>
<td>25.98</td>
<td>700</td>
<td>39.85</td>
<td>49.29</td>
<td>49.64</td>
<td>800</td>
<td>49.52</td>
<td>49.71</td>
</tr>
<tr>
<td>350</td>
<td>5.54</td>
<td>23.72</td>
<td>1100</td>
<td>39.95</td>
<td>49.31</td>
<td>49.66</td>
<td>1600</td>
<td>49.56</td>
<td>49.74</td>
</tr>
<tr>
<td>450</td>
<td>5.11</td>
<td>23.18</td>
<td>1400</td>
<td>39.99</td>
<td>49.33</td>
<td>49.69</td>
<td>3200</td>
<td>49.58</td>
<td>49.74</td>
</tr>
<tr>
<td>550</td>
<td>4.95</td>
<td>23.00</td>
<td>1700</td>
<td>40.65</td>
<td>49.35</td>
<td>49.70</td>
<td>6400</td>
<td>49.59</td>
<td>49.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K</th>
<th>Connect (%)</th>
<th>Accidents (%)</th>
<th>K (%)</th>
<th>Pumsb (%)</th>
<th>Re1 (%)</th>
<th>Retail (%)</th>
<th>K (%)</th>
<th>Wap (%)</th>
<th>La12 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.03</td>
<td>2.84</td>
<td>100</td>
<td>12.73</td>
<td>120.48</td>
<td>337.69</td>
<td>200</td>
<td>289.29</td>
<td>709.16</td>
</tr>
<tr>
<td>150</td>
<td>1.02</td>
<td>2.02</td>
<td>400</td>
<td>4.75</td>
<td>67.06</td>
<td>101.43</td>
<td>400</td>
<td>289.29</td>
<td>709.16</td>
</tr>
<tr>
<td>250</td>
<td>1.01</td>
<td>2.03</td>
<td>700</td>
<td>4.54</td>
<td>69.97</td>
<td>103.38</td>
<td>800</td>
<td>136.14</td>
<td>166.52</td>
</tr>
<tr>
<td>350</td>
<td>1.01</td>
<td>1.86</td>
<td>1100</td>
<td>4.61</td>
<td>72.08</td>
<td>113.24</td>
<td>1600</td>
<td>146.10</td>
<td>189.79</td>
</tr>
<tr>
<td>450</td>
<td>1.01</td>
<td>1.82</td>
<td>1400</td>
<td>4.64</td>
<td>74.33</td>
<td>118.65</td>
<td>3200</td>
<td>161.73</td>
<td>190.18</td>
</tr>
<tr>
<td>550</td>
<td>1.01</td>
<td>1.81</td>
<td>1700</td>
<td>4.98</td>
<td>76.63</td>
<td>120.80</td>
<td>6400</td>
<td>164.71</td>
<td>203.40</td>
</tr>
<tr>
<td>EASL</td>
<td>1.08</td>
<td>2.25</td>
<td>EASL</td>
<td>7.66</td>
<td>98.83</td>
<td>198.53</td>
<td>EASL</td>
<td>213.45</td>
<td>294.15</td>
</tr>
</tbody>
</table>

Table 7
The I/O saving ratio of TOP-MATA.

Table 8
The sharing length of various experimental data sets.

Fig. 15. TOP-DATA-R versus TOP-MATA: A Hard Disk Case.
Fig. 16. A comparison of computational performances.
Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (NSFC) (Nos. 70901002, 70890080, 90924020), National Science Foundation (NSF) via grant number CCF-1018151.

References

[15] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 1–12, Dallas, TX.
[34] H. Xiong, S. Shekhar, P.-N. Tan, V. Kumar, Exploiting a support-based upper bound of pearson’s correlation coefficient for efficiently identifying strongly correlated pairs, Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 334–343.