# EVALUATING COMPETITIVENESS FROM LARGE DATASETS BY MINING COMPETITORS

**Prof.D.Ravi, Ms.K.Sowmiya, Mr.U.Abhilash, Mr.K.Vignesh**

***Abstract***

*Success of any competitive business, is based on the ability to make an item more appealing to customers than the competition. Some tasks involve to form questions such that how do we quantify and formalize the competitiveness between two items? Who are the main competitors of a given item? What are the features and factors of an item that most affect its competitiveness? The impact and relevance of this problem to many domains, only a limited amount of work has been devoted toward an effective solution. Formal definition of the competitiveness between two items, based on the market segments that the both item can cover. Evaluation of competitiveness utilizes customer reviews, and wide range of information that is available in different domains. Efficient methods for evaluating competitiveness in large datasets from unstructured data sets and address the natural definition of finding the top-k competitors of a given item were evaluated. Finally, we evaluate and make customer to go through the expiry date of the product via notification from our site.*

*This system introduces an end-to-end methodology for mining such information from large data sets of customer reviews.*

***Manuscript received April, 2018.***

***Prof.D. Ravi****, Department of Computer Science & Engineering, Kathir college of Engineering, Coimbatore,Coimbatore, Tamilnadu, India*

***K.Sowmiya****, Department of Computer Science & Engineering, Kathir college of Engineering, Coimbatore, Coimbatore, Tamilnadu, India*

***Mr.U.Abhilash****, Department of Computer Science & Engineering, Kathir college of Engineering, Coimbatore,Coimbatore, Tamilnadu, India*

***Mr.K.Vignesh****, Department of Computer Science & Engineering, Kathir college of Engineering, Coimbatore,Coimbatore, Tamilnadu, India*

***Keywords****-Data mining, web mining, information search and retrieval, electronic commerce*

## 1. Introduction

The marketing and management community have focused on empirical methods for competitor identification. Due to the strategic importance of identifying and monitoring a firm's competitors, The marketing and management community have focused on empirical methods for identification of competitor. The competitiveness between two items is based on whether they compete for the attention and business of the same groups of customers. We specify all featured queries in textual form. The frequency of textual comparative evidence can vary greatly across domains. We propose a new formalization of the competitiveness between two items, based on the market segments that they can both cover. Extant research on the former has focused on mining comparative expressions (e.g., "Item A is better than Item B")
from the Web or other textual source. Even though such expressions can indeed be indicators
of competitiveness, they are absent in many domains.

- A formal definition of the competitiveness is, based on their appeal to the various customer in their market segments..
- A formal methodology for the identification of the different types

of customers in a given market segment, and also for the estimation of the percentage of customers that belong to each type.

- A highly scalable framework for finding the top-k competitors of a given item in vary from large unstructured datasets

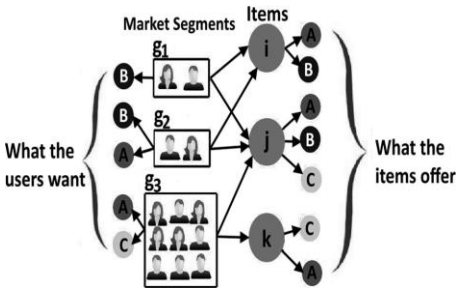


*Fig 1: Market Segment*

## 2. Proposed System

We describe a method for computing all the segments in a given market based on mining large review datasets. This method allows us to operationalize our definition of competitiveness and address the problem of finding the top-k competitors of an item in any given market. This problem presents significant computational challenges, especially in the presence of large datasets with hundreds or thousands of items, such as those that are often found in mainstream domains. We address these challenges via a highly scalable framework for top-k computation, including an efficient evaluation algorithm and an appropriate index. We propose a new concept for intimating the customers regarding the expiry date of the manufactured product, Where most of the consumer fails to notice.

We create a data collection of the customers and the product's details such as manufactured date, expiry date and the seller details. From the details collected we can give the notification message to the customer regarding the expiry date of the product prior to 10 days. We are also proposing the concept to raise complaints in web page regarding the expired products in a separate page so that we can take actions against the sellers.

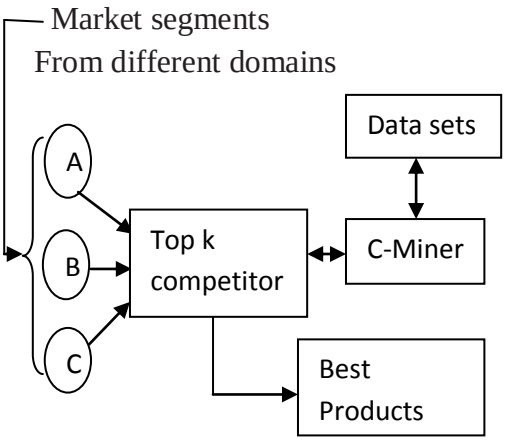


*Fig 2: System paradigm*

### 2.1 Defining Competitiveness

The typical user session on a review platform, such as Yelp, Amazon or Trip Advisor, consists of the following steps:

1) Specify all required features in a query.
2) Submit the query to the website's search engine and retrieve the matching items.
3) Process the reviews of the returned items and make a purchase decision.

In this setting, items that cover the user's requirements will be included in the search engine's response and will compete for her attention. On the other hand, non-covering items will not be considered by the user and, thus, will not have a chance to compete. Next, we present an example that extends this decision-making process to a multi-user setting.

Table 1

| ID | Segment Size | Features of Interest |
|---|---|---|
| $q_1$ | 100 | (parking, wi-fi) |
| $q_2$ | 50 | (parking) |
| $q_3$ | 60 | (wi-fi) |
| $q_4$ | 120 | (gym, wi-fi) |
| $q_5$ | 250 | (breakfast, parking) |
| $q_6$ | 80 | (gym, bar, breakfast) |

Each segment is represented by a query that includes the features that are of interest to the customers included in the segment. Information on each segment is provided in the provided Table

we define the competitiveness between I and j in a market with a feature subset F as follows:

$$C_f(i,j) = \sum_{q \epsilon 2^f} p(q) \times V_{i,j}^{q},$$

This definition has a clear probabilistic interpretation: given two items i; j, their competitiveness

CF (i; j) represents the probability that the two items are included in the consideration set of a random user. This new definition has direct implications for consumers, who often rely on recommendation systems to help them choose one of several candidate products.

## 2.2 Pairwise Coverage

We begin by defining the pairwise coverage of a single feature f. We then define the pairwise coverage of an entire query of features q.

**Pairwise Feature Coverage**:

We define the pairwise coverage V fi;j of a feature f by two items i; j as the percentage of all possible values of f that can be covered by both i and j. Formally, given the set of all possible values V for f,

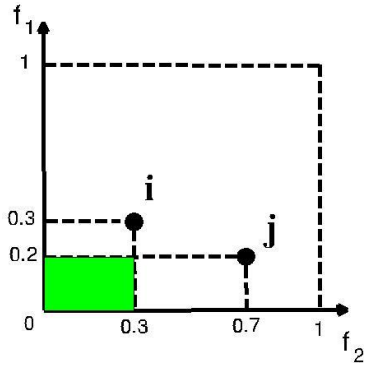


*Fig 3: Pair-wise Graph*

Two competitive items i and j, positioned according to their values for the two features:
f1½i_ ¼ 0:3; f2½i_ ¼ 0:3; f1½j_ ¼ 0:2, and f2½j_ ¼ 0:7.

We observe that the percentage of the 2-dimensional space that each item covers is equivalent to the area of the rectangle defined by the beginning of the two axes (0; 0) and the item's values for f1 and f2. For example, the covered area for item i is 0:3 _ 0:3 ¼ 0:09, equal to 9 percent of the entire space. Similarly, the pairwise coverage provided by both items is equal to 0:2 _ 0:3 ¼ 0:06 (i.e., 6 percent of the market).

## 2.3Estimating Query Probabilities

The definition of competitiveness is considered the probability p that a random customer will be represented by a specific query of features q, for every possible query. we describe how these probabilities can be estimated from real data. Feature queries are a direct representation of user preferences.

## 2.4 Extending Competitiveness Definition

Our competitiveness definition assumes that user requirements are uniformly distributed within the value space of each feature. This assumption allows us to build a computational model for competitiveness, but in practice it may not always be true. For instance, the number of users demanding quality in ½0; 0:1 might be different from those demanding a value in ½0:4; 0:5_. Moreover, for lack of more accurate information, it provides a conservative lower bound of our model's true effectiveness: having access to the distribution of interest within each feature could only improve the quality of our results

## 3.FINDING THE TOP-K COMPETITORS

***Problem 1 (Top-k Competitors Problem).***

We are presented with a market with a set of n items I and a set of features F. Then, given a single item i 2 I, we want to identify the k items from I that maximize $C_f(i,.)$.

A naive algorithm would compute the competitiveness between i and every possible candidate. The complexity of this brute force method is which, as we demonstrate in our experiments, is impractical for large datasets. One option could be to perform the naive computation in a distributed fashion. Even in this case, however, we would need one thread for each of the $n^2$ pairs. This is far from trivial, if one considers that n could measure in the tens of thousands. In addition, a naive Map Reduce implementation would face the bottleneck of passing everything through the reducer to account for the self-join included in the computation. In practice, the self-join would have to be implemented via a customized technique for reduce-side joins, which is a non-trivial and highly expensive operation

Conceptually, an item dominates another if it has better or equal values across features. At any item i that dominates j also achieves the maximum possible competitiveness with j, since it can cover the requirements of any customer covered by j.

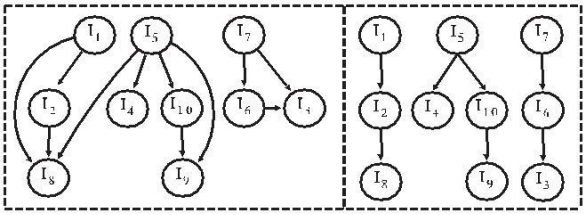

*Fig 4: The left side shows the dominance graph for a set of items. An edge Ii ! Ij means that Ii dominates Ij The right side of the figure shows the skyline pyramid.*

## 4. C-Miner Algorithm

*The C-Miner Algorithm*, an exact algorithm for finding the top-k competitors of a given item. This algorithm makes use of the skyline pyramid in order to reduce the number of items that need to be considered. Given that we only care about the top-k competitors, we can incrementally compute the score of each candidate and stop when it is guaranteed that the top-k have emerged

*Discussion of C-Miner*: The input includes the set of items I, the set of features F, the item of interest i, the number k of top competitors to retrieve, the set Q of queries and their probabilities, and the skyline pyramid DI . The algorithm first retrieves the items that dominate i,

These items have the maximum possible competitiveness with i. If at least k such items exist, we report those and conclude. Otherwise, we add them to Top K and decrement our budget of k accordingly The variable LB maintains the lowest lower bound from the current top-k set and is used to prune candidates. we initialize the set of candidates X as the union of items in the first layer of the pyramid and the set of items dominated by those already in the Top-K .Achieved calling GETSLAVES(TopK,$D_I$). Miner feeds the set of candidates X to the UPDATETOPK() routine, which prunes items based on the LB threshold. It then updates the Top-K set via the MERGE() function, which identifies the items with the highest competitiveness from Top K This can be achieved in linear time, since both X and Top K are sorted set to the worst (lowest) score among the new Top-K. Finally, GETSLAVES() is used to expand the set of candidates by including items that are dominated by those in X.

*Discussion of UPDATETOPK().*

This routine processes the candidates in X and finds atmost k candidates with the highest competitiveness with i. The routine utilizes a data structure local TopK, implemented as an associative array: the score of each candidate serves as the key, while its id serves as the value. The array is key-sorted, to facilitate the computation
of the k best items. The structure is automatically truncated so that it always contains at most k items. In lines 21-22
we initialize the lower and upper bounds. For every item j 2 X, low(j) maintains the current competitiveness score of j
as new queries are considered, and serves as a lower bound to the candidate's actual score. Each lower bound low(j) starts from 0, and after the completion of UPDATETOPK(), it includes the true competitiveness score CF

## 5.BOOSTING THE CMINER ALGORITHM

***Query Ordering***:

Our complexity analysis is based on the premise that C-Miner evaluates all queries Q for each candidate item j. However,this assumption naively ignores the algorithm's pruning ability, which is based on using lower and upper bounds on competitiveness scores to eliminate candidates early. Next, we show how to greatly improve the algorithm's pruning effectiveness by strategically selecting the processing order of queries (line 23 of C-Miner). C-Miner uses the following update rules for the lower and upper bounds for a candidate j

C-Miner uses the following update rules for the lower and upper bounds for a candidate j

## 6.Improving UPDATETOPK() and GETSLAVES()

In this section we describe several improvements to the C-Miner's two main routines. We implement all of these improvements into an enhanced algorithm, which we refer to as C-Miner++. We include this version in our experimental evaluation, where we compare its efficiency with that of C-Miner, as well as to that of other baselines. Even though C-Miner can effectively prune low quality candidates, a major bottleneck within the UPDATETOPK() function is the computation of the final competitiveness score between each candidate and the item of interest I. Speeding up this computation

can have a tremendous impact on the efficiency of our algorithm.
The GETSLAVES() method is used to extend the set of candidates by including the items that are dominated by those in a provided set (lines 7 and 15). Henceforth, we refer to this as the dominator set. A naive implementation would include

## 7. Experimental Evaluation

### *7.1 Datasets and Baselines*

Our experiments include four datasets, which were collected for the purposes of this project. The datasets were intentionally selected from different domains to portray the cross domain applicability of our approach. In addition to the full information on each item in our datasets, we also collected the full set of reviews that were available on the source website. These reviews were used to estimate queries probabilities, as described to extract the opinions of reviewers on specific features. The highly-cited method by Ding et al. [28] is used to convert each review to a vector of opinions, where each opinion is defined as a feature-polarity combination.

### *7.2 Evaluating Comparative Methods*

Previous work on competitor mining utilizes textual comparative evidence between two items. However, these approaches assume that such comparative evidence is abundant in the available data. In this experiment, we evaluate this assumption on our four datasets. For every pair of items in each dataset, we report the number of reviews that mention both items and the number of reviews that include a direct comparison between the two items. We extract such comparative evidence based on the union of "competitive evidence".

### *7.3 Computational Time*

In this experiment we compare the speed of C-Miner with that of the two baselines (Naive and G-Miner), as well as with that of the enhanced C-Miner++ algorithm. Specifically, we use each algorithm to compute the set of top-k competitors for each item in our datasets. Each plot reports the average time, in seconds, per item (y-axis) against the various k values (x-axis).

### *7.4 Ordering Efficiency*

The COV ordering scheme, which determines the processing order of queries by C-Miner. Next, we demonstrate COV's superiority over the PINC and P-DCR ordering schemes, which process queries in increasing and decreasing probability order, respectively. For each approach, we compute the number of pairwise query coverages V q i;j that need to be computed (line 25 of Algorithm 1). We present the results in Fig. 6. The x-axis of each plot holds the value of k, while the y-axis holds the average number of processed queries/coverages. We observe a consistent advantage for COV, across datasets and values of k. This verifies that COV increases the efficiency of C-Miner by rapidly eliminating candidates that fail to cover important queries.

## 8. RELATED WORK

This paper significantly extends our preliminary work on Competitiveness. A is better than Item B" "or item A versus Item B" is indicative of their competitiveness. However, as we have already discussed in the introduction, such evidence is typically scarce or even non-existent in many

mainstream domains. As a result, the applicability of such approaches is greatly limited. We provide empirical evidence on the sparsity of co-occurrence information in our experimental evaluation.

Skyline Computation. Our work leverages concepts and techniques from the literature on skyline computation and has ties to the recent publications in reverse skyline queries Even though the focus of our work is different, we intend to utilize the advances in this field to improve our framework in future work.

## 9.CONCLUSION

We presented a formal definition of competitiveness between two items, which we validated both quantitatively and qualitatively. Our formalization is applicable across domains, overcoming the shortcomings of previous approaches. We consider a number of factors that have been largely overlooked in the past, such as the position of the items in the multi-dimensional feature space and the preferences and opinions of the users. Our work introduces an end-to-end methodology for mining such information from large datasets of customer reviews. Based on our competitiveness definition, we addressed the computationally challenging problem of finding the top-k competitors of a given item. The proposed framework is efficient and applicable to domains with very large populations of items. The efficiency of our methodology was verified via an experimental evaluation on real datasets from different domains.

## REFERENCES

[1] R. Deshpand and H. Gatingon, "Competitive analysis," Marketing Lett., vol. 5, pp. 271–287, 1994.

[2] W. T. Few, "Managerial competitor identification: Integrating the categorization, economic and organizational identity perspectives," Joseph M. Katz Graduate School of Business, doctoral dissertaion, University of Pittsburgh, Pittsburgh, PA, USA, 2007.

[3] M. Bergen and M. A. Peteraf, "Competitor identification and competitor analysis: A broad-based managerial approach," Managerial Decision Econ., vol. 23, pp. 157–169, 2002.

[4] Z. Ma, G. Pant, and O. R. L. Sheng, "Mining competitor relationships from online news: A network-based approach," Electron. Commerce Res. Appl., vol. 10, pp. 418–427, 2011.

[5] G. Pant and O. R. L. Sheng, "Avoiding the blind spots: Competitor identification using web text and linkage structure," in Proc. Int.Conf. Inf. Syst., 2009, Art. no. 57.

[6] K. Lerman, S. Blair-Goldensohn, and R. McDonald, "Sentiment summarization: Evaluating and learning user preferences," in Proc. 12th Conf. Eur. Chapter Assoc. Comput. Linguistics, 2009,pp. 514–522.

[7] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: A survey," ACM SIGMOD Rec., vol. 40, no. 4, pp. 11–20, 2012.