Profit Mining: From Patterns to Actions [⋆]

Ke Wang[1], Senqiang Zhou[1], and Jiawei Han[2]

[1] Simon Fraser University
{wangk,szhou}@cs.sfu.ca
[2] University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Abstract. A major obstacle in data mining applications is the gap between the statistic-based pattern extraction and the value-based decision making. We present a *profit mining* approach to reduce this gap. In profit mining, we are given a set of past transactions and pre-selected target items, and we like to build a model for recommending target items and promotion strategies to new customers, with the goal of maximizing the net profit. We identify several issues in profit mining and propose solutions. We evaluate the effectiveness of this approach using data sets of a wide range of characteristics.

1 Introduction

Data management today is required of the ability to extract interesting patterns from large and raw data to help decision making, i.e., *data mining*. Often, patterns are deemed "interesting" on the basis of passing certain statistical tests such as support/confidence [AIS93,AMSTV96,AS94]. To an enterprise, however, it remains unclear how such patterns can be used to maximize a business objective. For example, knowing association rules $\{Perfume\} \rightarrow Lipstick$, $\{Perfume\} \rightarrow Diamond$, ... that are related to $Perfume$, a store manager wishing to maximize the profit margin still cannot tell which of *Lipstick*, *Diamond*, ..., and what price, should be recommended to a customer buying $Perfume$. Simply recommending all items will overwhelm the customer and defeat the purpose of recommendation. Simply recommending the most profitable item, say *Diamond*, or the most likely item, say *Lipstick*, does not maximize the profit because there is often an inverse correlation between the likelihood to buy and the dollar amount to spend. The major obstacle lies at the gap between *individual, statistic-based summaries* extracted by traditional rule mining and a *global, profit-driven action* required by business decision making.

1.1 The profit mining problem

We propose the *profit mining* approach to address this issue. In profit mining, we are given a collection of past transactions, target items and non-target items,

[⋆] Research was supported in part by research grants from the Natural Science and Engineering Research Council of Canada.

and promotion codes containing the pricing and cost information of items. A transaction contains one target sale of the form $\langle I, P, Q \rangle$, for some target item I, and several non-target sales of the form $\langle I', P, Q \rangle$, for non-target items I'. The presence of $\langle I, P, Q \rangle$ (or $\langle I', P, Q \rangle$) in a transaction conveys that I (or I') was sold in the quantity of Q under the promotion code P. *Profit mining* is to build a model, called the *recommender*, that recommends a pair of target item I and promotion code P to future customers whenever they buy non-target items. A successful recommendation generates $(Price(P) - Cost(P)) \times Q$ *profit*, where $Price(P)$ and $Cost(P)$ are the price and cost represented by P, and Q is the quantity sold because of the recommendation. The goal is to maximize the total profit of target items on future customers.

Unlike a basic prediction model that "repeats the past", profit mining is expected to "get smarter from the past". An example illustrates the point. Suppose that 100 customers each bought 1 pack of *Egg* at the pack price of \$1/pack, and another 100 customers each bought one package of 4-pack at the package price of \$3.2/4-pack. Assume that each pack costs \$0.5 in both cases. The first 100 customers generate the profit of $100 \times (1 - 0.5) = \$50$, and the second 100 customers generate the profit of $100 \times (3.2 - 2) = \$120$. The total profit is \$170. With no inherent difference between the two groups of customers, to the next 200 new customers a basic prediction model will recommend the pack price in one half case and the package price in the other half case. This will repeat the profit of \$170. In contrast, profit mining is expected to reveal that the profit has increased at the package price and recommend this price to all the next 200 customers. This will generate the profit of \$240.

A "quick solution" to profit mining is to find several most probable recommendations using a basic prediction model, and re-rank them by taking into account both probability and profit. In this solution, the profit is considered as an afterthought. For example, for the decision tree [Q93] (as the basic predication model), whether a rule is extracted as a pattern or is pruned as a noise is solely based on the frequency information. The study in [MS96] shows that pushing the profit objective into model building is a significant win over the afterthought strategy.

1.2 The issues

The key to profit mining is to recommend "right" items and "right" prices. If the price is too high, the customer will go away without generating any profit; if the price is too low or if the item is not profitable, the profit will not be maximized. Our approach is to exploit data mining to extract the patterns for right items and right prices. Let us examine the issues/requirements in this context.

1. **Profit-based patterns**. A pure statistic-based approach will favor the rule $\{Perfume\} \rightarrow$ *Lipstick* because of higher confidence, and a pure profit-based approach will favor rule $\{Perfume\} \rightarrow$ *Diamond* because of higher profit. Neither necessarily maximizes the profit. Indeed, items of high profit are often statistically insignificant because fewer people buy expensive stuffs.

To maximize profit, the rule extraction needs to take into account both statistical significance and profit significance.

2. **Shopping on unavailability**. To maximize profit, it is important to recognize that paying a higher price does not imply that the customer will not pay a lower price; rather, it is because no lower price was available at the transaction time. This behavior is called *shopping on unavailability*. Taking into account this behavior in rule extraction will bring new opportunities for increasing the profit.
3. **Explosive search space**. A typical application has thousands of items and much more sales, any combination of which could be a trigger of recommendation. Any table-based representation such as decision tree and neural network require thousands of columns or input nodes to encode items and sales. To make matters worse, patterns are often searched at alternative concepts (e.g., food, meat, etc.) and prices of items.
4. **Optimality of recommendation**. Given the large search space, finding the exact optimal recommender is infeasible. Still, it is important, both theoretically and practically, to obtain some optimality guarantee within some important classes of recommenders.
5. **Interpretability of recommendation**. It is highly desirable or even necessary to have a recommender that is able to explain the rationale of recommendation in a human understandable way. For example, knowing what triggers the recommendation of certain target items could be useful for setting up a cross-selling plan.

1.3 Our approaches

To address the scalability and interpretability in Requirements 3 and 5, we exploit association rules [AIS93,AS94] for constructing the recommender. By some extension of association rules, we are able to incorporate the customer preference into the mining. This addresses Requirement 2. However, association rules alone do not maximize the profit because they are not profit-sensitive and do not optimize a global objective. One technical contribution of this work is to combine individually extracted association rules into a single model that maximizes the projected profit on future customers. The novelty of this approach is the selection of association rules based on both statistical significance and profit significance. We show that the recommender constructed is optimal within an important class of recommenders. Experiments show encouraging results.

In Section 2, we define the profit mining problem. In Section 3, we construct a recommender using extended association rules. In Section 4, we simplify the recommender to increase the projected profit on future transactions. In Section 5, we study the effectiveness of the approach. We review related work in Section 6. Finally, we conclude the paper.

2 Problem definition

In profit mining, we like to promote the sales of *target items* based on the sales of *non-target items*. Every item has one or more *promotion codes*. A promotion codes contain the price and cost information for a promotion package. A *target sale* (resp. *non-target sale*) has the form $\langle I, P, Q\rangle$, representing a sale of quantity Q of item I under promotion code P. A *transaction*, written as $\{s_1, \ldots, s_k, s\}$, consists of one target sale s and several non-target sales $s_1, \ldots, s_k$. A *recommender* recommends a target item I and a promotion code P (of I) to future customers whenever they buy some non-target items. If the recommendation is successful, i.e., the customer actually buys some quantity Q of item I under P, it generates $(Price(P) - Cost(P)) \times Q$ profit, where $Price(P)$ and $Cost(P)$ are the price and cost represented by P.

Example 1. Suppose that an item $2\%\_Milk$ *has four promotion codes (not necessarily offered at the same time): (\$3.2/4-pack,\$2), (\$3.0/4-pack,\$1.8), (\$1.2/pack, \$0.5), and (\$1/pack, \$0.5), where the first element denotes the price and the second element denotes the cost. Let P denote (\$3.2/4-pack,\$2). A sale* $\langle Egg, P, 5\rangle$ *generates of* $5 \times (3.2 - 2) = \$6$ *profit. Note that the price, cost and quantity in a sale refer to the same packing (e.g., 4-pack).*

Some (descriptive) items, such as $Gender = Male$, do not have a natural notion of promotion code. For such items, we set $Price(P)$ and Q to 1 and $Cost(P)$ to 0, and the notion of profit becomes the notion of support. In this paper, we assume that all target items have a natural notion of promotion code.

Definition 1 (Profit mining). Given a collection of past transactions (over some specified target and non-target items), the problem of *profit mining* is to find a recommender that generates as much profit as possible on target items over future transactions. ∇

This problem implicitly assumes that the given transactions are representative in recommendation structure of the entire population. To define the profit mining problem precisely, we need to specify the representation of recommenders. Our first consideration is that recommendation often depends on some categories (or concepts) of items. The categorization of items can be specified by a concept hierarchy [HF95,SA95].

A *concept hierarchy* , denoted H, is a rooted, directed acyclic graph, with each leaf node representing an item and a non-leaf node representing a concept. For example, assume that an item $Flake\_Chicken$ belongs to categories $Chicken$, $Meat$, $Food$, ANY. If a customer bought $Flake\_Chicken$, obviously the customer also "bought" $Chicken$, $Meat$, $Food$, Any. For non-target items, such generalization allows us to search for the best category that capture certain recommendations. We do not consider categories for target items because it does not make sense to recommend a concept and a price (such as $Applicance$ for\$100) unless the concept refers to a specific item known to the customer. Therefore, in the concept hierarchy, target items are (immediate) children of the root ANY.

Our second consideration has to do with the *shopping on unavailability* mentioned in Introduction. Consider a customer who has bought one 2-pack of Egg for 3.80$. If the lower price 3.50$/2-pack has been offered (for the same item) before his/her purchase, clearly the customer would have taken the offer, even though there is a mismatch in the two prices. This suggests that the acceptance of recommendation should be based on the "intention" of customers, rather than on the exact match of price. We say that a promotion code P is *more favorable than* a promotion code P', denoted $P \prec P'$, if P offers more value (to the customer) for the same or lower price, or offers a lower price for the same or more value, than P' does.

For example, \$3.50/2-pack offers a lower price than \$3.80/2-pack for the same value, and \$3.50/2-pack offers more value than \$3.50/1-pack for the same price. In both cases, the former is more favorable than the latter. However, \$3.80/2-pack is not (always) more favorable than \$3.50/pack because it is not favorable to pay more for unwanted quantity.

Mining on availability - MOA. If a customer is willing to buy an item under some promotion code, we assume that the customer will buy the item under a more favorable promotion code. This assumption is called the *mining on availability*, or simply *MOA*. To incorporate the knowledge of MOA into search, we treat a more favorable promotion code P as a "concept" of a less favorable one P'. The effect is that a sale under P' implies a sale under P. This can be done by extending the concept hierarchy H as follows.

Definition 2 (MOA(H)). For each item I, let $(\prec, I)$ denote the hierarchy of pairs $\langle I, P\rangle$ induced by $\prec$ on the promotion codes P for I, with I being added as the root. $MOA(H)$ is the hierarchy obtained by making each leaf node I in H as the root of the hierarchy $(\prec, I)$. ∇

A transaction is generalized by generalizing its sales using $MOA(H)$ as described below.

Definition 3 (Generalized sales). In $MOA(H)$, (i) every parent node is a *generalized sale* of every child node; (ii) every node of the form $\langle I, P\rangle$ is a *generalized sale* of a sale of the form $\langle I, P, Q\rangle$; (iii) "is a generalized sale of" is transitive. A set of generalized sales $G = \{g_1, \ldots, g_k\}$ *matches* a set of sales $S = \{s_1, \ldots, s_p\}$ if each g_i is a generalized sale of some s_j. ∇

(i) generalizes a sale using concepts and favorable promotion codes. (ii) generalizes a sale by ignoring the quantity of the sale. A generalized sale has one of the forms $\langle I, P\rangle$, or I, or C, where P is a promotion code, I is an item, C is a concept. For a target item I, we consider only generalized sales of the form $\langle I, P\rangle$ because only this form represents our recommendation of a pair of target item and promotion code. Note that a generalized sale of the form $\langle I, P\rangle$ contains the packing quantity defined by the promotion code P. The quantity of individual sales will be factored in the profit of rules.

Example 2. Consider a non-target item Flaked_Chicken, abbreviated as FC, and a target item Sunchip. Figure 1(a) shows the concept hierarchy H. Suppose that FC has three promotion codes: \$3, \$3.5, and \$3.8, and Sunchip has *three promotion codes: \$3.8, \$4.5, and \$5. For simplicity, we omit the cost and assume that the packing quantity for all promotion codes is 1. Figure 1(b) shows $MOA(H)$. $\langle FC, \$3.8\rangle$ and its ancestors are generalized sales of sales $\langle FC, \$3.8, Q\rangle$. $\langle FC, \$3.5\rangle$ and its ancestors are generalized sales of sales $\langle FC, \$3.5, Q\rangle$ or $\langle FC, \$3.8, Q\rangle$. $\langle FC, \$3\rangle$ and its ancestors are generalized sales of sales $\langle FC, \$3, Q\rangle$, or $\langle FC, \$3.5, Q\rangle$, or $\langle FC, \$3.8, Q\rangle$. Similar generalization exists for target item Sunchip.*

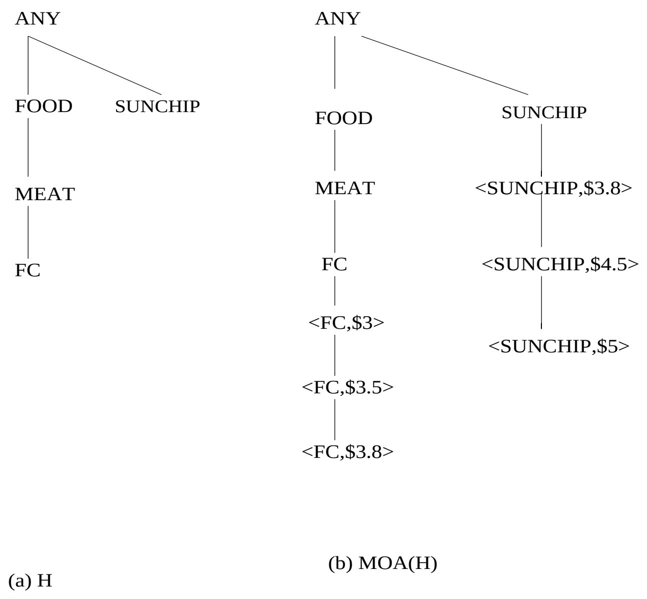


Fig. 1. H and $MOA(H)$

Definition 4 (Recommenders). A *rule* has the form $\{g_1, \ldots, g_k\} \rightarrow \langle I, P\rangle$, where $g_1, \ldots, g_k$ are generalized non-target sales such that no g_i is a generalized sale of other g_j, and $\langle I, P\rangle$ is a generalized target sale. Consider a customer represented by a set of non-target sales $\{s_1, \ldots, s_p\}$. A rule $\{g_1, \ldots, g_k\} \rightarrow \langle I, P\rangle$ *matches* the customer if $\{g_1, \ldots, g_k\}$ generalizes $\{s_1, \ldots, s_p\}$. If a matching rule $\{g_1, \ldots, g_k\} \rightarrow \langle I, P\rangle$ is selected to make recommendation and if the customer buys some quantity Q of I under the recommended promotion code, the profit generated is $(Price(P) - Cost(P)) \times Q$. A *recommender* is a set of rules plus a method for selecting rules to make recommendation to future customers. ∇

Note that the condition that no g_i is a generalized sale of other g_j implies that $k \leq p$. In the above framework, each transaction contains one target sale and each rule recommends one pair of target item and promotion code. To apply to transactions containing several target sales and recommendation of several pairs of target item and promotion code, we can generate the same type of rules but we select several rules for each recommendation. The number of rules selected per recommendation can be specified by the user. Therefore, our framework is not a restriction.

3 Constructing the initial recommender

We construct a recommender in two steps. In the first step, we generate association rules and specify the method for selecting recommendation rules. This recommender does not necessarily produce a high profit on future transactions because many rules are too specific. In the second step, we remove such rules on a basis of increasing the projected profit on future customers. This section focuses on the first step.

3.1 Generating rules

Since the space of candidate rules is extremely large, we must focus on rules of some minimum "worth". The worth is a measure of how well a rule captures the "customer intention". Suppose that a rule $r : \{g_1, \ldots, g_k\} \rightarrow \langle I, P \rangle$ matches a given transaction $t : \{s_1, \ldots, s_p, \langle I_t, P_t, Q_t \rangle\}$, where $\langle I_t, P_t, Q_t \rangle$ is the target sale. If $\langle I, P \rangle$ generalizes $\langle I_t, P_t, Q_t \rangle$, that is, $I = I_t$ and $P \prec P_t$, then r has captured the intention of t. In this case, we credit the worth of r by the profit of r generated on t. To estimate this profit, we regard t as a future customer and determine the quantity Q the customer will buy under the more favorable promotion code P. The *generated profit* of r on t is defined as

- $p(r,t) = (Price(P) - Cost(P)) \times Q$, if $\langle I, P \rangle$ generalizes $\langle I_t, P_t, Q_t \rangle$, or
- $p(r,t) = 0$, otherwise.

We consider two methods of estimating the actual purchase quantity Q for the more favorable promotion code P under MOA. **Saving MOA** assumes that the customer keeps the original quantity Q_t unchanged, thus, saving money. **Buying MOA** assumes that the customer keeps the original spending $P_t \times Q_t$ unchanged, thus, increasing the quantity to $Q = P_t \times Q_t / P$. Both assumptions are conservative by not increasing the spending at a favorable promotion. These assumptions are favorable to the customer in that the customer either spends less for the same quantity or spends the same for more quantity. A more greedy estimation could associate the increase of spending with the relative favorability of P over P_t and the uncertainty of customer behaviors. We will consider such estimation in our experiments.

Definition 5 (Worth of a rule). For any set of generalized sales X, let $Supp(X)$ denote the percentage of the transactions matched by X. Consider a rule $G \rightarrow g$.

- $Supp(G \rightarrow g)$: The *support* of $G \rightarrow g$, defined as $Supp(G \cup \{g\})$.

- $Conf(G \rightarrow g)$: The *confidence* of $G \rightarrow g$, defined as $Supp(G \cup \{g\})/Supp(G)$.
- $Prof_{ru}(G \rightarrow g)$: The *rule profit* of $G \rightarrow g$, defined as $\Sigma_t p(G \rightarrow g, t)$, where t is a transaction matched by $G \rightarrow g$.
- $Prof_{re}(G \rightarrow g)$: The *recommendation profit* of $G \rightarrow g$, defined as $Prof_{ru}(G \rightarrow g)/N$, where N is the number of transactions matched by $G \rightarrow g$. ∇

The recommendation profit is on a per-recommendation basis and factors in both the hit rate (i.e., confidence) and the profit of the recommended item. It is possible that a rule of high recommendation profit matches only a small number of transactions that have large profit. Determining whether such rules should be used is a tricky issue and will be examined in Section 4.

To find rules of minimum worth, the user can specify minimum thresholds on these measures. The minimum support must be specified to take advantage of the support-based pruning [AS94]. If all target items have non-negative profit, a similar pruning is possible for rule profit and the minimum support can be replaced with the minimum rule profit. We follow [SA95,HF95] to find association rules, with $MOA(H)$ being the input concept hierarchy.

In the rest of discussion, let $\mathcal{R}$ denotes the set of rules generated as above, plus the *default rule* $\emptyset \rightarrow g$, where g is the generalized target sale that maximizes $Prof_{re}(\emptyset \rightarrow g)$. Adding the default rule ensures that any set of non-target sales has at least one matching rule in $\mathcal{R}$.

3.2 Specifying recommendation rules

A key for making recommendation is to select a recommendation rule from $\mathcal{R}$ for a given customer. Our selection criterion is maximizing the recommendation profit of the selected rule, as stated below.

Definition 6 (Coverage of rules). Let *body*(r) denote the set of generalized sales in the body of r, and let $|body(r)|$ denote the number of such generalized sales. For any two rules r and r', we say that r is *ranked higher than* r'

- (Profit per recommendation) if $Prof_{re}(r) > Prof_{re}(r')$, or
- (Generality) if $Prof_{re}(r) = Prof_{re}(r')$, but $Supp(r) > Supp(r')$, or
- (Simplicity) if $Supp(r) = Supp(r')$, but $|body(r)| < |body(r')|$, or
- (Totality of order) if $|body(r)| = |body(r')|$, but r is generated before r',

in that order. Given a set B of non-target sales, a rule r in $\mathcal{R}$ is the *recommendation rule* for B if r matches B and has highest possible rank. This is called the *most-profitable-first* selection, or *MPF*. We also say that recommendation rule r *covers* B. ∇

Confidence is not mentioned here because it is indirectly factored in the recommendation profit.

Definition 7 (MPF recommender). The *MPF recommender* is the set of rules $\mathcal{R}$ plus the MPF for recommendation rules. ∇

4 Optimizing the MPF recommender

However, the MPF does not deal with the overfitting of rules because a high recommendation profit does not imply a high support. It does not work to simply remove rules of low support by a high minimum support because high-profit items typically have a low support. Our approach is to prune rules on the basis of increasing the *projected profit* on future customers: Suppose that we know how to estimate the projected profit of a rule r using the given transactions covered by r, denoted by $Cover(r)$. We can prune one rule at a time if doing so increases the projected profit of the recommender, defined as the sum of the projected profit of all rules in the recommender. This approach must answer the following questions:

- Question 1: If some rule is pruned, which remaining rules will cover those transactions that were previously covered by the pruned rule? This information is necessary for subsequent prunings.
- Question 2: How do we select the rule for pruning at each step? Does the pruning order matter? Does such pruning produce an "optimal" recommender?
- Question 3: How do we estimate the projected profit of a rule?

We answer these questions in the rest of this section.

4.1 The covering relationship

If a rule is pruned, we choose the "next best" rule to take over the coverage of its transactions. To define this notion of "next best", we say that r is *more general than* r', or r' is *more special than* r, if *body*(r) generalizes *body*(r'). r and r' do not necessarily have the same head. If a rule is more special and ranked lower than some other rule in $\mathcal{R}$, this rule will never be used as a recommendation rule because some general rule of a higher rank will cover whatever it matches. From now on, we assume that all such rules are removed from $\mathcal{R}$.

Definition 8. In the *covering tree* of $\mathcal{R}$, denoted $\mathcal{CT}$, a rule r is the *parent* of a rule r' if r is more general than r' and has the highest possible rank. If a rule r is pruned, the parent of r will cover the transactions covered by r ∇

In $\mathcal{CT}$, rules are increasingly more specific and ranked higher walking down the tree. Therefore, it makes sense to prune specific rules in the bottom-up order. The effect is to "cut off" some subtrees to maximize the projected profit.

4.2 The cut-optimal recommender

By "cutting off" the subtree at a rule r, r becomes a leaf node and covers all the transactions previously covered by (the rules in) the subtree, according to the covering tree. The "best cut" should yield the maximum projected profit, defined below.

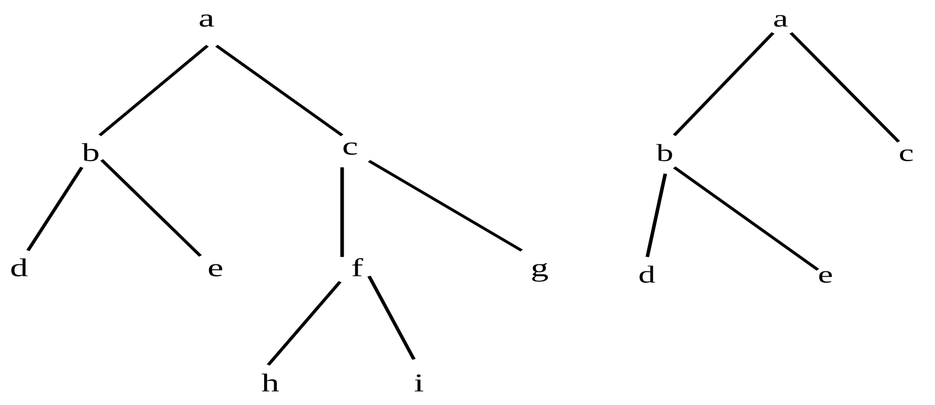


Fig. 2. Cuts

Definition 9. A *cut* of $\mathcal{CT}$ contains exactly one node on each root-to-leaf path in $\mathcal{CT}$. For a cut C, $\mathcal{CT}_C$ denotes the tree obtained from $\mathcal{CT}$ by pruning all subtrees at the nodes in C. A cut C is *optimal* if $\mathcal{CT}_C$ has the maximum (estimated) projected profit and C is as small as possible. $\mathcal{CT}_C$ is the *cut-optimal* recommender if C is an optimal cut. ∇

The essence of cuts is that either all children are pruned or none is pruned. By cutting the tree this way, we avoid the sensitivity caused by different orders of pruning child nodes. Consider the covering tree $\mathcal{CT}$ on the left in Figure 2. Examples of cuts are $\{a\}$, $\{b, c\}$, $\{d, e, c\}$, $\{b, f, g\}$, $\{b, h, i, g\}$, $\{d, e, f, g\}$, $\{d, e, h, i, g\}$. $\{a, b\}$ and $\{d, e, f\}$ are not a cut because they contain two nodes or no node on some root-to-leaf path. On the right in Figure 2 is $\mathcal{CT}_C$ for $C = \{d, e, c\}$.

We can show that the optimal cut is unique.

Theorem 1. A covering tree has exactly one optimal cut (therefore, exactly one cut-optimal recommender). ∇

Proof: If there are two optimal cuts C_1 and C_2, they must cross over in the sense that some portion of C_1, say $\overline{C_1}$, is above C_2, and some portion of C_2, say $\overline{C_2}$, is above C_2. Then it can be shown that the smaller cut formed by $\overline{C_1}$ and $\overline{C_2}$ yields a recommender that has no less projected profit than $\mathcal{CT}_{C_1}$ or $\mathcal{CT}_{C_2}$, contradicting that C_i's are optimal cuts. ∇

To find the optimal cut of $\mathcal{CT}$, one can start with the root of $\mathcal{CT}$ and consider every possible subset of child nodes for expanding the current node, and for each subset considered, recursively repeat this expansion for each node in the subset. This method examines an exponential number of subsets of child nodes and is not scalable for large recommenders. We present a linear time algorithm for finding the optimal cut of a given covering tree.

Finding the cut-optimal recommender. The algorithm finds the optimal cut of $\mathcal{CT}$ in a bottom-up traversal of the tree. Consider the current non-leaf node r. Let $Tree\_Prof(r)$ denote the projected profit of the subtree at r.

Let $Leaf\_Prof(r)$ denote the projected profit of r as a leaf node. The estimation of of these profits will be explained shortly. Intuitively, $Tree\_Prof(r)$ and $Leaf\_Prof(r)$ are the projected profit before and after pruning the subtree at r. If $Leaf\_Prof(r) \leq Tree\_Prof(r)$, we prune the subtree at r immediately; otherwise, we do nothing at r. "Pruning the subtree at r" means deleting the nodes and edges below r and modifying the coverage $Cover(r)$ to contain all transactions previously covered by the nodes in the subtree. After all nodes are traversed, the unpruned top portion of $\mathcal{CT}$ is the cut-optimal recommender, as stated in Theorem 2 below.

Theorem 2. The recommender $\mathcal{CT}$ at the end of the above bottom-up traversal is cut-optimal. ∇

Proof: Note that the pruning at each node does not affect the pruning at other nodes. Therefore, if the test at a node is in favor of pruning the subtree at the node to increase the projected profit of the recommender, an optimal cut will never contain this subtree. ∇

Now we sketch the idea of estimating the projected profit of a rule r, denoted $Prof_{pr}(r)$. We estimate $Prof_{pr}(r)$ by $X \times Y$. X is the (estimated) # of "hits" of r, i.e., # of acceptances of the recommendation, in a random population of $N = |Cover(r)|$ customers that are covered by r. Y is the observed average profit per hit. We compute X using the *pessimistic estimation* borrowed from [CP34,Q93]: Suppose that E of the N transactions covered by r are not hit by the recommendation of r, i.e., do not match the righ-hand side of r. If this is regarded as a sample of a binomial distribution over the entire population of transactions, for a given confidence level CF the upper limit of the probability of non-hit in the entire population is estimated by $U_{CF}(N, E)$ as computed in [CP34,Q93]. Then, $X = N \times (1 - U_{CF}(N, E))$. Y is estimated by

$$\frac{\Sigma_{t \in Cover(r)} p(r, t)}{\#\text{ of hits in } Cover(r)}. \tag{1}$$

Recall that $p(r, t)$ is the generated profit of r on transaction t (defined in Section 2). $Tree\_Prof(r)$ is computed as the sum of $Prof_{pr}(u)$ over all nodes u in the subtree at r. This sum can be computed incrementally in the bottom-up traversal of the tree. $Leaf\_Prof(r)$ is computed as $Prof_{pr}(r)$ by assuming that r covers all the transactions covered by the subtree at r.

5 Evaluation

We like to validate two claims: the cut-optimal recommender is profitable, and incorporating profit and MOA into model building is essential for achieving this profitability.

5.1 The methodology

We perform 5 runs on each dataset using the 5-fold cross-validation. In particular, the dataset is divided into 5 partitions of equal size, and each run holds back one (distinct) partition for validating the model and uses the other 4 partitions for building the model. The average result of the 5 runs is reported. We define the *gain* of a recommender as the ratio of generated profit over the recorded profit in the validating transactions in the held-back partition:

$$\Sigma_t p(r,t)/\Sigma_t \text{ the recorded profit in } t$$

where $p(r,t)$ (defined in Section 2) is the generated profit of the recommendation rule r on a validating transaction t. The higher the gain, the more profitable the recommender. For saving MOA and buying MOA, the maximum gain is 1 because the spending is never increased under a favorable promotion code. Unless otherwise stated, saving MOA is the default. (The gain for buying MOA will be higher if all target items have non-negative profit.)

PROF+MOA represents the cut-optimal recommender, emphasizing that both profit and MOA are used in building the recommender. We compare PROF+MOA with:

– PROF–MOA: the cut-optimal recommender without MOA. This comparison will reveal the effectiveness of MOA.
– CONF+MOA: the cut-optimal recommender using the binary profit: $p(r,t) = 1$ if the recommendation is a hit; otherwise, $p(r,t) = 0$. Thus, the model building ignores the profit and relies on the hit rate (i.e., confidence) for ranking and pruning rules. This comparison will reveal the effectiveness of profit-based model building.
– CONF–MOA: CONF+MOA without MOA.
– kNN: the *k-nearest neighbor* classifier [YP97]. Given a set of non-target sales, kNN selects k transactions (the k nearest neighbors), for some fixed integer k, that are most similar to the given non-target sales and recommends the pair of target item and promotion code most "voted" by these transactions. We used the kNN that is tailored to sparse data, as in [YP97] for classifying text documents, and we applied MOA to tell whether a recommendation is a hit. These modifications substantially increase the hit rate and profit.
– MPI: the *most profitable item* approach, which simply recommends the pair of target item and promotion code that has generated most profit in past transactions.

5.2 Datasets

The synthetic datasets were generated by the IBM synthetic data generator [1], but modified to have price and cost for each item in a transaction. First, we apply the IBM generator to generate a set of transactions, with the number of transactions $|T| = 100K$ and the number of items $|I| = 1000$, and default settings for other parameters. Items are numbered from 1 to $|I|$. For simplicity, each item has a single cost and a single packing for all promotion codes. In this case, we use "price" for "promotion code". The cost of item i is denoted by $Cost(i)$. For

[1] http://www.almaden.ibm.com/cs/quest/syndata.html#assocSynData

item i, we generate the cost $Cost(i) = c/i$, where c is the maximum cost of a single item, and m prices $P_j = (1 + j \times \delta)Cost(i)$, $j = 1, \ldots, m$. We use $m = 4$ and $\delta = 10\%$. Thus, the profit of item i at its price P_j is $j \times \delta \times Cost(i)$. Each item in a transaction is mapped to a non-target sale by randomly selecting one price from the m prices of the item. For simplicity, all sales have unit quantity.

We consider two distributions for generating the target sale in each transaction. In dataset I, we consider two target items with cost of \$2 and \$10 respectively. Many important decision makings such as *direct marketing* [MS96] are in the form of two-target recommendation. We model the sales distribution of the two target items using the Zipf law [2]: the target item of cost \$2 occurs five times as frequently in the dataset as the target item of cost \$10. Therefore, the higher the cost, the fewer the sales. The price generation and selection for target items are similar to those for non-target items. In dataset II, there are 10 target items, numbered from 1 to 10. The cost of target item i is $Cost(i) = 10 \times i$. Unlike dataset I, the frequency of target items follows the normal distribution [3]: most customers buy target items with the cost around the mean. Figure 3(e) and Figure 4(e) show the profit distribution of target sales in dataset I and dataset II.

5.3 Results

Figure 3 shows that the result on dataset I. Figure 3(a) shows the gain of the six recommenders (for kNN, $k = 5$ gives the best result) with two obvious trends: PROF+MOA performs significantly better than other recommenders, and the recommenders with MOA perform significantly better than their counterparts without MOA. This clearly demonstrates the effectiveness of incorporating profit and MOA into the search of recommenders. PROF+MOA achieves 76% gain at minimum support 0.1%. This gain is encouraging because the saving MOA adopted is conservative in profit estimation. Interestingly, the curve for PROF−MOA shows that profit-based mining is not effective without MOA, and the curves for CONF+MOA shows that MOA is not effective either without profit-based mining.

To model that a customer buys and spends more at a more favorable price, for each validating transaction, we compare the recommended price P_p with the recorded price P_q of the target item. Recall that $P_j = (1 + j \times \delta)Cost(i)$, $j = 1, \ldots, 4$, for item i. If $q - p = 1$ or $q - p = 2$, that is, the recommended price P_p is 1 or 2 step lower than the recorded price P_q, we assume that the customer doubles the purchase quantity in the transaction with the probability of 30%. We denote this setting by $(x = 2, y = 30\%)$. If $q - p = 3$ or $q - p = 4$, we assume that the customer triples the purchase quantity in the transaction with the probability of 40%. We denote this setting by $(x = 3, y = 40\%)$. Figure 3(b) shows the gain of all recommenders using MOA with the purchase quantity determined

[2] http://alexia.lis.uiuc.edu/ standrfr/zipf.html

[3] see http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm, for example

by ($x = 2, y = 30\%$) and ($x = 3, y = 40\%$). With this more realistic shopping behavior, the gain for all recommenders increases. PROF+MOA with the setting ($x = 3, y = 40\%$), denoted PROF(x=3,y=40%), achieves the encouraging gain of 2.23 (at minimum support of 0.1%)!

Figure 3(c), which uses the legend in Figure 3(a), shows the hit rate of recommenders. PROF+MOA and CONF+MOA achieve the hit rate of 95%. For minimum support of 0.08%, Figure 3(d) shows the hit rate at different profit ranges. "Low", "Medium", and "High" represent the lower, middle, and higher 1/3 of the maximum profit of a single recommendation. The legend from top to bottom corresponds to left to right in the bar chart. For example, kNN has nearly 100% hit rate at the "Low" range, but less than 10% at the "High" range. CONF+MOA and CONF–MOA also have a similar trend. In contrast, PROF+MOA is "profit smart" in maintaining a high hit rate in a high profit range. Though MPI picks up the hit rate in a high profit range, the hit rate is still too low compared to PROF+MOA. PROF–MOA is unstable for this dataset.

Figure 3(f), which uses the legend in Figure 3(a), shows the number of rules in recommenders. kNN and MPI have no model, so no curve is shown. The number of rules prior to the cut-optimal phase (not shown here) is typically several hundreds times the final number. This shows that the pruning method proposed effectively improves the interpretability of recommenders. MOA generally increases the size due to additional rules for alternative prices. Not surprisingly, the minimum support has a major impact of the size. The execution time is dominated by the step of generating association rules. In our experiments, we adopted the multi-level association rules mining whose performance has been studied elsewhere [SA95,HF95]. The time for constructing the covering tree from generated association rules and for the bottom-up traversal is insignificant.

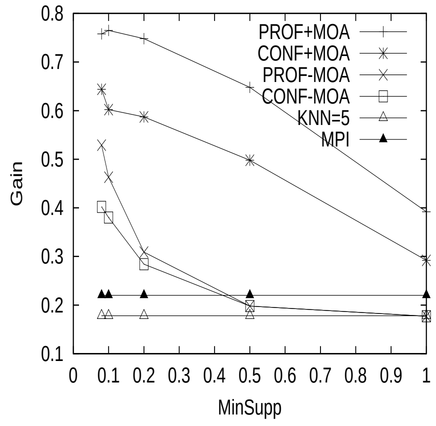
Figure 4 shows the result on dataset II. This dataset has 40 item/price pairs for recommendation because each target item has 4 prices. Therefore, the random hit rate is 1/40, which is more challenging than dataset I. Despite the difference in cost distribution and a lower hit rate, the result is consistent with that of dataset I, that is, supports the effectiveness of profit-based mining and MOA.

We also modified kNN to recommend the item/price of the most profit in the k nearest neighbors. This is a post-processing approach because the profit is considered only after the k nearest neighbors are determined. For dataset I, the gain increases by about 2%, and for dataset II, the gain decreases by about 5% (not shown in the figure). Thus, the post-processing does not improve much.

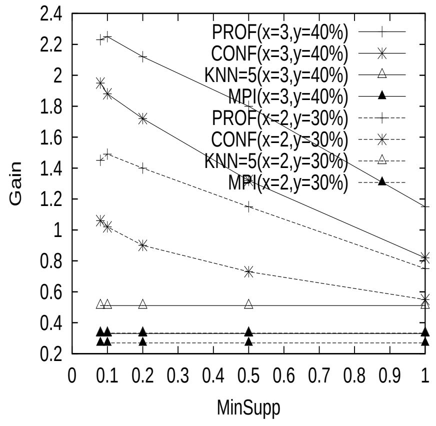
In summary, the experiments confirm our goals set at the beginning of the section.

6 Related work

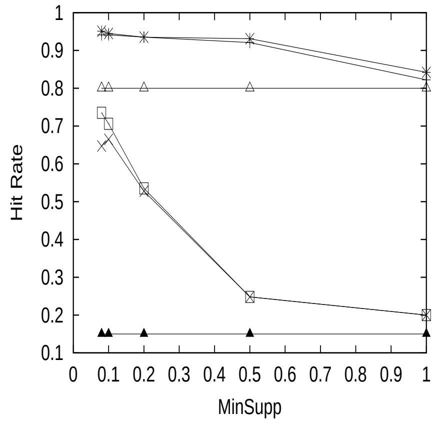
[MS96] considers the customer value while training a neural network. As discussed in Introduction, neural network does not scale up for sparse data and large databases, does not easily incorporate domain knowledge, and does not produce an understandable model. [BSVW99] considered the problem of stock-



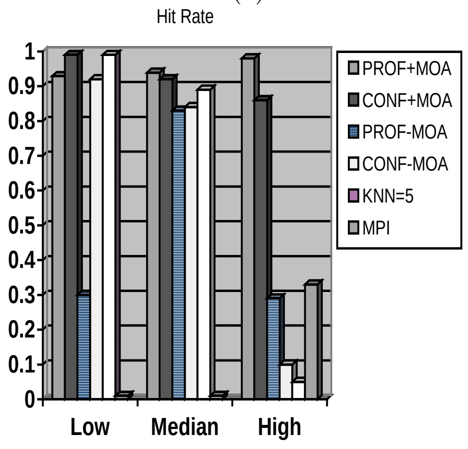
(a)



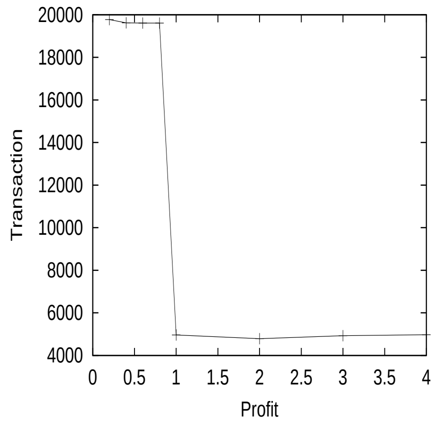
(b)



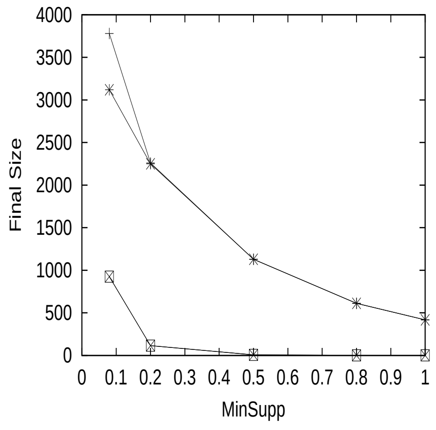
(c)



(d)

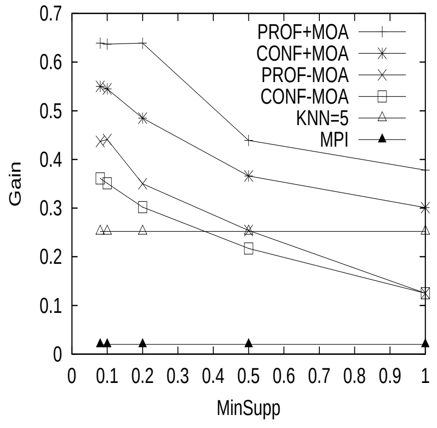


(e)

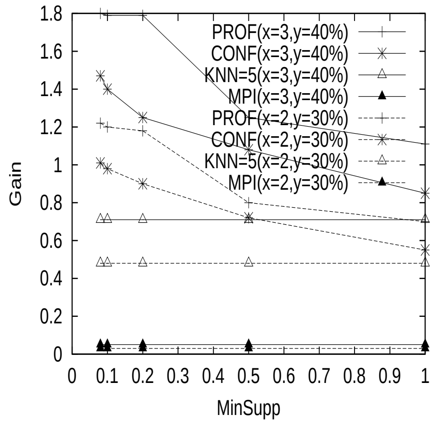


(f)

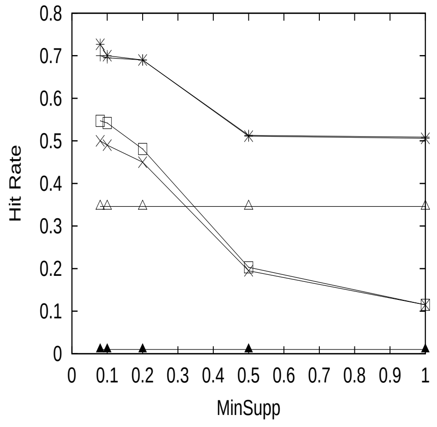
Fig. 3. The result for dataset I



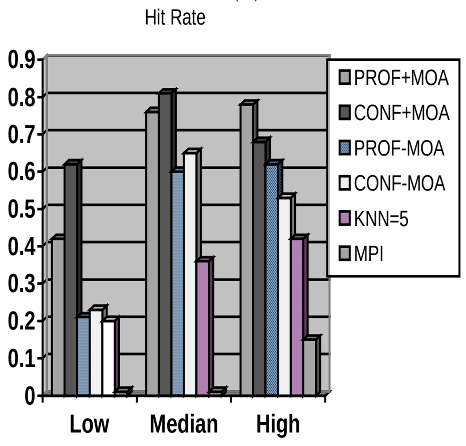
(a)



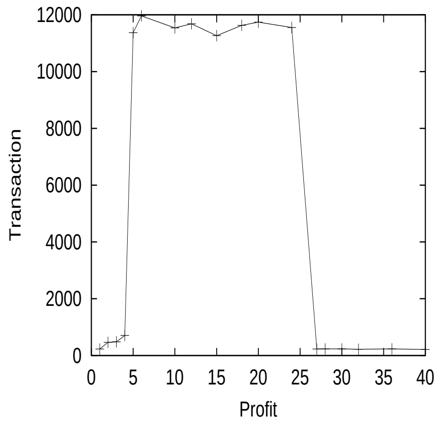
(b)



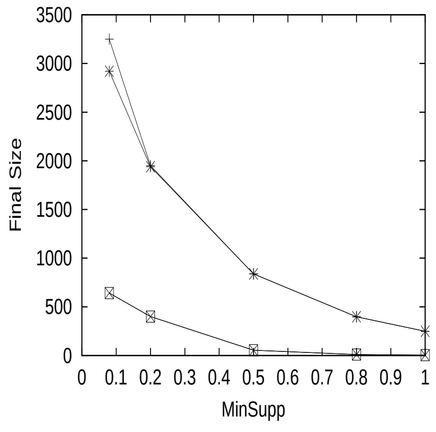
(c)



(d)



(e)



(f)

Fig. 4. The result for dataset II

ing a profitable set of items in supermarket environments. In profit mining, we recommend items and promotion codes on a per-customer basis, not on a per-problem basis like in the stocking problem. Indeed, a solution to the stocking problem does not give a clue on how to make such recommendations.

Our work is similar in motivation to the *actionability* of patterns [ST96] - the ability of the pattern to suggest concrete and profitable action by the decision-makers. Recently, Kleinberg el at presented the *microeconomic view* of data mining [KPR98]. The microeconomic view approach is to $max_{x \in D} \Sigma_{i \in C} g(x, y_i)$, where $g(x, y_i)$ is the "utility" of a decision x on a given customer i. In profit mining, we are to $max_{x \in D} g(x, C)$, where g is the total profit (a kind of utility) of a recommender x on *future* customers, given the data about *current* customers C which is a sample of the entire population. In the latter case, the model building has to tell whether a pattern is too specific for the entire population.

Our work benefits from the scalability of mining *(generalized) association rules* for large databases [AIS93,AS94,SA95,HF95]. However, association rules neither address the economic value of transactions nor produce a global action plan for decision making.

The *cost-sensitive classification* [P99] assumes an error metric of misclassification and minimizes the error on new cases. No such error metric is given in profit mining. Rather, we assume that customers spend some money on recommended items at recommended prices, and we maximize the profit by recommending right items and prices. It does not work to map each item/price recommendation to a class because the sales quantity, which obviously affects the profit, is not factored. More fundamentally, the cost-sensitive classification follows the trend of the (historical) data as correctly as possible (with respect to the given error metric), whereas profit mining may depart from the trend, if necessary, to increase the profit, as explained in Introduction.

Collaborative filtering [RV97] makes recommendation to a customer by aggregating the "opinions" (such as rating about movies) of a few "advisors" who share the same taste with the customer. The goal is to maximize the hit rate of recommendation. For items of varied profit, maximizing profit is quite different from maximizing hit rate. Collaborative filtering relies on carefully selected "item endorsements" for similarity computation, and a good set of "advisors" to offer opinions. Such data are not easy to obtain. The ability of recommending prices, in addition to items, is another major difference between profit mining and other recommender systems.

7 Conclusion

We presented a profit-based data mining called *profit mining*. The goal of profit mining is to construct a recommender that recommends target items and promotion codes on the basis of maximizing the profit of target sales on future customers. We presented a scalable construction of recommenders to address several important requirements in profit mining: pruning specific rules on a profit-sensitive basis, dealing with the behavior of shopping on unavailability, dealing with sparse and explosive search space, ensuring optimality and interpretability of recommenders. Experiments on a wide range of data characteristics show very encouraging results. The novelty of this research is extracting patterns on a profit-sensitive basis and combining them into a global actionable plan for decision making. This economic orientation and actionability will contribute to wider and faster deployment of data mining technologies in real life applications.

References

[AIS93] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large datasets. SIGMOD 1993, 207-216

[AMSTV96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo. Fast discovery of association rules. Advances in knowledge discovery and data mining, 307-328, AAAI/MIT Press, 1996

[AS94] R. Agrawal and R. Srikant. Fast algorithm for mining association rules. VLDB 1994, 487-499

[BSVW99] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: a case study. KDD 1999, 254-260

[CP34] C.J. Clopper and E.S. Pearson. The use of confidence or Fiducial limits illustrated in the case of the binomial. Biometrika, 26:4, Dec. 1934, 404-413. Also available from http://www.jstor.org/journals/bio.html

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB 1995, 420-431

[KPR98] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. Journal of Knowledge Discovery and Data Mining, 1998, vol.2, 311-324 (also http://www.cs.berkeley.edu/ christos/dm1.ps)

[MS96] B. Masand and G. P. Shapiro. A comparison of approaches for maximizing business payoff of prediction models. KDD 1996, 195-201

[P99] P. Domingos. MetaCost: a general method for making classifiers cost-sensitive. KDD 1999, 155-164

[Q93] J.R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann, 1993

[RV97] P. Resnick and H.R. Varian, Eds. CACM special issue on recommender systems. *Communications of the ACM*, Vol. 40, No. 3, 56-58, 1997

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. VLDB 1995, 407- 419

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interresting in knowledge discovery systems. IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, 1996

[YP97] Y. Yang and J.O. Pederson. A comparative study on feature selection in text categorization. International Conference on Machine Learning 1997