Chapter 2

Overview

The generation of refinement candidates is based on co-occurrence between keywords in document databases, which is a common principle of query refinement researches in document retrieval. The focus of data mining approach is also based on finding relationships among data items. In this chapter, this approach of query refinement by combining traditional ones using co-occurrence and data mining approach, is overviewed. The main efforts are as listed in the following and are explained in detail in the sequel sections.

1. Introducing AR into query refinement, meanwhile taking the effect of screening into consideration.

2. Using *stem rule* to reduce refinement candidates by pruning derivable ARs.

3. Using *coverage* to guarantee that no documents be automatically discarded by system.

2.1 Association Rule

As shown in [Agra93, Fayy96, Han95, Sava95, Srik96], association rules (ARs) in data mining have been mainly used for discovering rules in large databases such as sale transaction databases. An AR is described as the follows. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items, and let $T = \{t_1, t_2, ..., t_n\}$ be a set of transactions each of which contains a subset of items I. An AR is an implication of form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The intuitive meaning of such a rule is that transactions in the database which contains the items in X tend to also contain the items in Y. The confidence (denoted by Cnf) of a rule $X \Rightarrow Y$ is the percentage of the transactions which contain the items in X also contain the items in Y. The support (denoted by Spt) of a rule $X \Rightarrow Y$ is the percentage of the transactions that contain both the items in X and the items in Y. An example of such a rule might be that 80% of people who buy jackets also buy shoes. Given two thresholds: minimum support (denoted by $MinSpt$) and minimum confidence(denoted by $MinCnf$), mining ARs is to find any rule which satisfies $Spt \geq MinSpt$ and $Cnf \geq MinCnf$.

Table 2.1 is an example of transaction database D, where $I = \{i_1, i_2, ..., i_6\}$ is the set of items and $T = \{t_1, t_2, ..., t_6\}$ is the set of transactions. $i_1 \Rightarrow i_6$ with Spt=50% and Cnf=75% is an AR when $MinSpt$ is set to 30% and $MinCnf$ is set to 50% .

Table 2.1: An example of transaction database

T	Items					
t_1	i_1		i_3			
t_2	i_1	i_2				i_6
t_3		i_2	i_3	i_4		
t_4	i_1					i_6
t_5		i_2		i_4	i_5	
t_6	i_1	i_2			i_5	i_6

[Liu98a] and [Liu98b] presented a query refinement support method for a document retrieval system in which ARs among keywords mined in large document databases are used. Let q, p be sets of keywords. In this context, an AR of $q \Rightarrow p$ means that documents containing all of the keywords in q will also contain all the keywords in p, with support and confidence. If 200 documents include set of keywords q and 50 documents include both sets of keywords q and p, then, we say that the support of rule $q \Rightarrow p$ is 50 documents and the confidence of it is 50/200=0.25. ARs are based on a statistical method in which the support makes rules meet certain frequency and the confidence describes degree of relation. The confidence also indicates that relationship is not symmetric, as confidences of $q \Rightarrow p$ and $q \Rightarrow p$ are often different from each other. It is very im-

Table 2.2: A example of document database

D	Keyword					
d_1	k_1		k_3			
d_2	k_1	k_2				k_6
d_3		k_2	k_3	k_4		
d_4	k_1					k_6
d_5		k_2		k_4	k_5	
d_6	k_1	k_2			k_5	k_6

portant that, unlike the similarity and co-occurrence, association rules are not symmetric.

In the following, we discuss the query refinement based on ARs. Let q be an user's query, and $q \Rightarrow p$ be an AR with respect to the query q. When the retrieval result set $D(q)$ of the query q is very large, the set of keywords p may be conjunctively added to q to form a new query $q' = q \cup p$. The set of keywords p is called refinement candidate. $Spt(q \Rightarrow p) = |D(q')|$ is the number of documents retrieved by the new query q'. And $Cnf(q \Rightarrow p) = |D(q')|/|D(q)| < 1$ is rate of reducing result set. In Table 2.2, if the original query is $q = \{k_1\}$ then documents $D(q) = \{d_1, d_2, d_4, d_6\}$ will be retrieved by q. When $MinSpt$ is set to 2 and $MinCnf$ is set to 0.5 then $k_1 \Rightarrow k_6$ with Spt=3 and Cnf=0.75 is an AR and a new query $q' = \{k_1, k_6\}$ based on $k_1 \Rightarrow k_6$ retrieves documents $D(q') = \{d_2, d_4, d_6\}$, which is a subset of $D(q)$. In this example such a k_6 is called a refinement candidate of query $q = \{k_1\}$. In the meanwhile, $k_1 \Rightarrow k_2$ with Spt=2 and Cnf=0.5 is another AR and k_2 is also a refinement candidate of query $q = \{k_1\}$. A query has more than one refinement candidate. In the query refinement support, a set of refinement candidates of a query is generated from all the rules with respect to the query. The following list is all the ARs whose left hand side is $\{k_1\}$ when $MinSpt$ is set to 2 and $MinCnf$ is set to 0.5.

$k_1 \Rightarrow k_6$, Spt=3, Cnf=0.75

$k_1 \Rightarrow k_2$, Spt=2, Cnf=0.5

$k_1 \Rightarrow k_2k_6$ Spt=2, Cnf=0.5

$C(q) = \{\{k_2\}, \{k_6\}, \{k_2, k_6\}\}$ is a set of refinement candidates based on these ARs with respect to query $q = \{k_1\}$.

Figure 2.1 illustrates an overview of this approach. First, the keywords will be extracted from a document database. ARs are mined for those keywords which have a good screening effectiveness. In the example above, k_2 has better screening effectiveness then k_6. ARs are stored and managed in a rule base. Second, the system selects ARs from the rule base with respect to the user's query. The ARs will be displayed including the supports and the

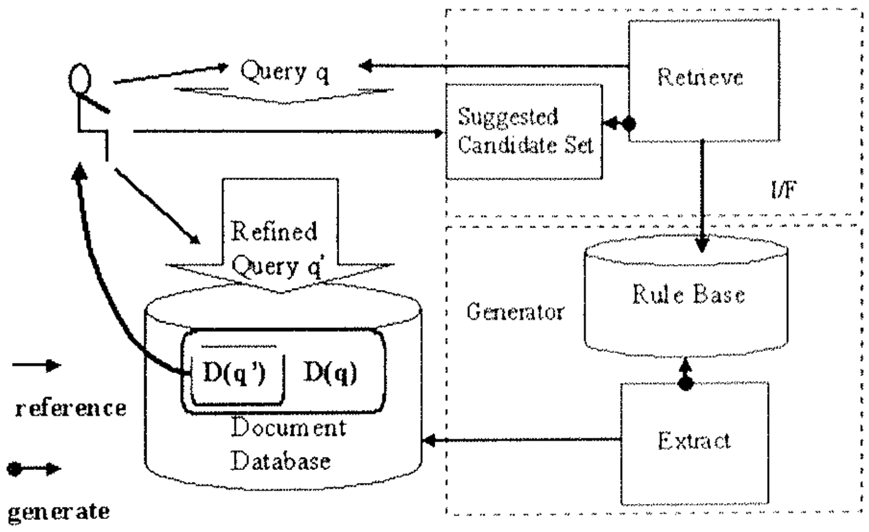

Figure 2.1: An overview of query refinement support system

confidences. Third, the user who submits a query can refine her/his original query based on the ARs being displayed. This process will be carried out repeatedly until the user successfully refines user's query.

The three issues are mainly concerned in this thesis as the following sections.

2.2 Generation of Association Rules for Query Refinement

Given a query containing a set of keywords q. When an AR $q \Rightarrow p$ with a high confidence is picked up, the keywords included in p will be added to a submitted query q. However, when the AR has a high confidence, the effectiveness of screening of the number of documents being retrieved is very small.

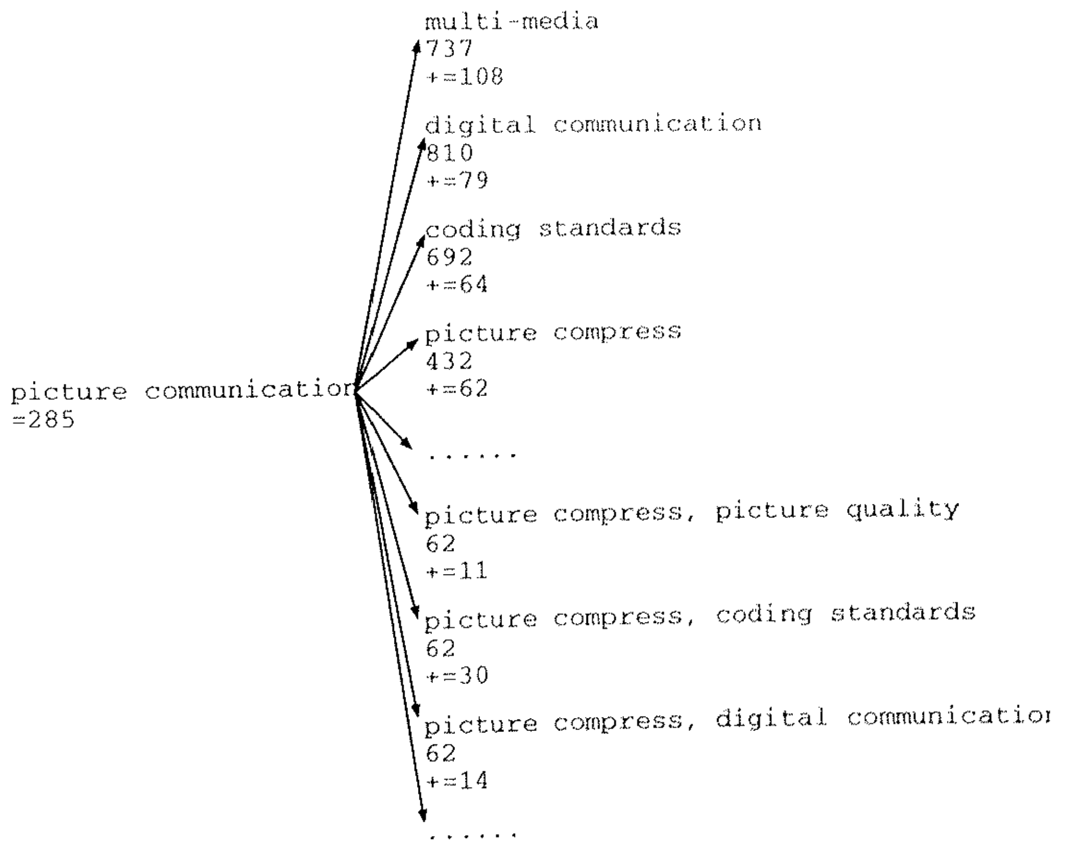

Figure 2.2: An example of query refinement

In Table 2.2, let k_6 be an original query that retrieves documents $D(q) = \{d_2, d_4, d_6\}$. $k_6 \Rightarrow k_1$ with Spt=3 and Cnf=1.0 is an AR when $MinSpt$ is set to 2 and $MinCnf$ is set to 0.5. A new query $q = \{k_6, k_1\}$ based on the AR $k_6 \Rightarrow k_1$ retrieval documents $D(q) = \{d_2, d_2, d_6\}$. The result sets $D(q)$ and $D(q')$ is the same, which means that $k_6 \Rightarrow k_1$ is not an effective AR for query refinement.

In the real example of figure 2.2, suppose that an user wants to retrieve documents on the topics of transmitting of image. Using a keyword "picture communication", this user can get 285 hits. If user's query is refined by adding "multimedia", then the number of documents being retrieved is 108. It can be seen that adding keyword "multimedia" can only screen little because the confidence is high between these two keywords. On the contrary, the keywords, "compression" and "digital", are always associated to "image". By adding these two keywords into the original query, the number of documents being retrieved is screened to 14.

In order to obtain better screening effectiveness, maximum confidence is proposed to be used instead of minimum confidence.

2.3 Reduction of Refinement Candidates

In figure 2.3, the number of refinement candidates having only one keyword is 225, 124, 50 when minimum support is set to 3, 5, 10 respectively. If the refinement candidates that have more than one keywords are also considered, the number of all refinement candidates will be very large. This may cause the confusion for users to browse a large list of refinement candidates. It is also difficult that huge rules are generated and stored. If the number of keywords in a document database is k, then a naive method must check $k \times (k-1)$ combinations to generate all ARs between two keywords, and so on. Finally, it must check all $k!$ combinations for generating ARs.

To avoid a large number of rules being generated, *stem rules* are introduced from which the other association rules can be derived. The stem rules are the only rules that we need to store in the rule base. All the other applicable association rules can be generated from the stem rules at run time instead of being generated from scratch.

According to stem rule, displaying rules is realized by stepwise method. Figure 2.3 shows a structure of rules. Comparing figure 2.3 with figure 2.2, it can be found that the number of refinement candidates displayed to users one time is reduced.

The number of refinement candidates may also be reduced by setting minimum support to a large value. It is noticed that the effect of stem rule reducing the number of refinement candidates is under the condition of certain minimum support. Moreover, setting minimum support to a large value might cause the problem of coverage that will be discussed in next section.

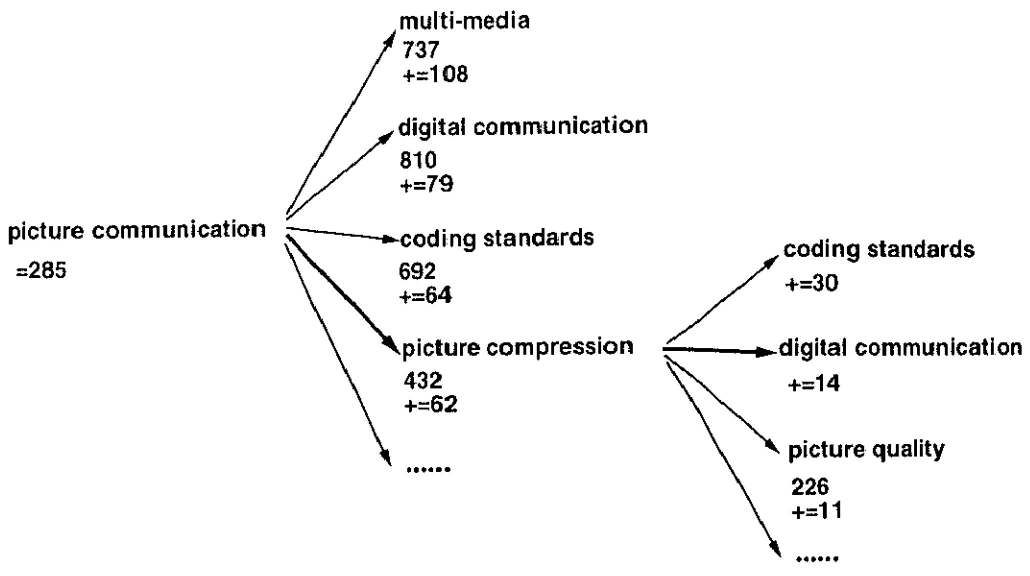

Figure 2.3: An example of stem rules

2.4 Coverage

Most of query refinement systems use the thresholds to limit size of candidates or number of documents related a candidate. However, using threshold would often cause that set of candidates can not cover the original query.

For example, in Table 2.2, let $q = \{k_1\}$ be the original query, then documents $D(q) = \{d_1, d_2, d_4, d_6\}$ will be retrieved. When minimum support is set to 2 and minimum confidence is set to 0.5, the set of refinement candidates is $C(q) = \{\{k_2\}, \{k_6\}, \{k_2, k_6\}\}$.

Hence, q can be refined to $\{k_1, k_2\}$, $\{k_1, k_6\}$ or $\{k_1, k_2, k_6\}$. The sets of documents retrieved by each of these refined queries are shown in the following.

$$D(\{k_1, k_2\}) = \{d_2, d_6\}.$$
$$D(\{k_1, k_6\}) = \{d_2, d_4, d_6\}.$$
$$D(\{k_1, k_2, k_6\}) = \{d_2, d_6\}.$$

If user refines his/her query q by referring to it, the document d_1 of $D(q)$ does not have chance to be retrieved. This is called the problem of "coverage". A set of keywords is said to be a coverage of query if the documents retrieved by the query can be retrieved using the same set of keywords. With the concept of stem rules and coverage, the number of ARs can be reduced to a feasible level, and it can ensure that all documents can be covered. This research proposes a method of coverage that solves the problem of coverage mentioned above.