Row-Pruning Algorithm Tutorial

Dr. Edel Garcia
admin@miislita.com

Abstract: This is a short tutorial on the row-pruning algorithm (RPA) and its unconditional version (URPA). These algorithms identify the largest answer sets associated to a term sequence family. These are discussed in *IR Watch – The Newsletter* (IRW-2007-8), *Association Rules Theory – Part 2*.

Keywords: row-pruning algorithm, unconditional pruning, RPA, URPA, association rules

Background: Let C be a collection of documents and let T = {t_1, t_2… t_m} be *m* unique terms extracted from C. Assume that term *i* is combined with all other terms such that starting with term *i* a family of term sequences is obtained. Assume that these term sequences are hidden (latent) in C. The purpose of RPA and URPA is to identify the composition of these term sequences and to find those that occur more frequently in C.

Pruning is accomplished by evaluating the condition ***…If conf >= minconf Then…*** where ***conf*** is an experimental confidence value and ***minconf*** is a threshold confidence value. This is aimed at identifying which association rules are not likely to occur by chance. The ***If-then*** condition can also be evaluated in terms of support values, but in such cases one would be evaluating the reliability of a rule.

In RPA the ***…If conf >= minconf Then…*** condition is evaluated at the level of parent-child rows (pc-RPA) or at the level of parent rows (p-RPA). In URPA the condition is not evaluated at either level. However, after identifying all candidate sequences the user has the option of evaluating the condition at least once.

Since URPA retains more candidate sequences than RPA, association rules that might be missing with RPA can be found with URPA. On the other extreme, one may end evaluating more number of association rules with poor confidence. URPA and RPA have been compared in IRW-2007-08.

Applications: In addition to document collections, row-pruning algorithms can be applied to the analysis of search logs or to retail transactions wherein customers tend to buy items in a strict order. An example follows.

Suppose that a customer buys item **a** and, due to this transaction, he receives by mail a discount coupon referencing the transaction with an offer to buy item **b**. He goes back to the store, buys **b**, and also buys item **c**. The purchase of **a** and then of **b** are not entirely independent from one another, but can be tied to a common transaction ID, so as **c**. These purchases can be described by an association rule of the form

{a b} → {c}

that reads

"*Customers buying **a** and then **b** tend to buy next **c***"

which is the same as saying:

*"Customers tend to buy **c** after buying **a** and then **b**"*

Similar rules can be derived from spatio-temporal data or in some cases from data derived from compulsive customers. Thus, finding association rules from purchases conducted in a strict order within a time window can provide retailers with new opportunities for cross-selling other products or services.

Problem: The following terms were extracted from a document collection (C): *mortgage, loan, refinance, equity,* and *rates*. Use URPA to find the term sequences that occur more frequently in C. Assume that C is Google or a suitable search engine.

Solution: Let a = mortgage, b = loan, c = refinance, d = equity, and e = rates.

Step 1. Construct a term-term matrix with ij rows populated with answer set counts for the corresponding ij pairs. Each row of this array is termed a parent row. Essentially you need to search the term(s) in EXACT mode in Google and record the number of documents retrieved.

{X}	{a}	{b}	{c}	{d}	{e}
{a}	-	{ab}	{ac}	{ad}	{ae}
{b}	{ba}	-	{bc}	{bd}	{be}
{c}	{ca}	{cb}	-	{cd}	{ce}
{d}	{da}	{db}	{dc}	-	{de}
{e}	{ea}	{eb}	{ec}	{ed}	-

Figure 1. A term-term array with five parent rows.

For instance, the following results were obtained for the first parent row.

{X}	{a}	{b}	{c}	{d}	{e}
{a}	-	{ab}	{ac}	{ad}	{ae}

Candidate terms	{a } = {mortgage}	{b} = {loan}	{c} = {refinance}	{d} = {equity}	{e} = {rates}
Candidate term counts	{a} = 178000000	{b} = 163000000	{c} = 47900000	{d} = 176000000	{e} = 476000000
{X}	{a}	{b}	{c}	{d}	{e}
{a}	x	{a b} = 1880000	{a c} = 1910000	{a d} = 404000	{a e} = 22200000

Figure 2. Answer sets for the first parent row shown in Figure 1.

Step 2. Identify from the parent row the sequence that generates the largest answer set and ignore all other sequences. Next, recombine the identified sequence with remaining candidate terms. This generates a child row. From this row identify the child sequence that generates the largest answer set and again, ignore all other sequences from this row. Recombine the retained sequence with remaining candidate terms. Iterate all these steps until all sequences of the family are identified. For instance, the following results were obtained by querying Google. Retained sequences are colored in red.

	Candidate terms	{a } = {mortgage}	{b} = {loan}	{c} = {refinance}	{d} = {equity}	{e} = {rates}
	Candidate term counts	{a} = 178000000	{b} = 163000000	{c} = 47900000	{d} = 176000000	{e} = 476000000
Parent Row 1 Family	{X}	{a}	{b}	{c}	{d}	{e}
{k1}	{a}	x	{a b} = 1880000	{a c} = 1910000	{a d} = 404000	{a e} = 22200000
{k1 k2}	{a e}	x	[a e b} = 27700	{a e c} = 150000	{a e d} = 967	x
{k1 k2 k3}	{a e c}	x	{a e c b} = 59600	x	{a e c d} =14500	x
{k1 k2 k3 k4}	{a e c b}	x	x	x	{a e c b d} = 0	x
{k1 k2 k3 k4 k5}	{a e c b d}					

Figure 3. EXCEL data for the family of term sequences obtained from *mortgage*.

From the first parent row we can see that *mortgage* gives birth to the following family of term sequences.

k1 = *mortgage*
"k1 k2" = "*mortgage rates*"
"k1 k2 k3" = "*mortgage rates refinance*"
"k1 k2 k3 k4" = "*mortgage rates refinance loan*"
"k1 k2 k3 k4 k5" = "*mortgage rates refinance loan equity*"

The last sequence with {a e c b d} = 0 is included for illustration purposes.

The following Venn Diagram illustrates a typical term sequence family like the one we have just identified. Note that finding these families is equivalent to conducting a greedy binary partition on the **largest** answer sets.

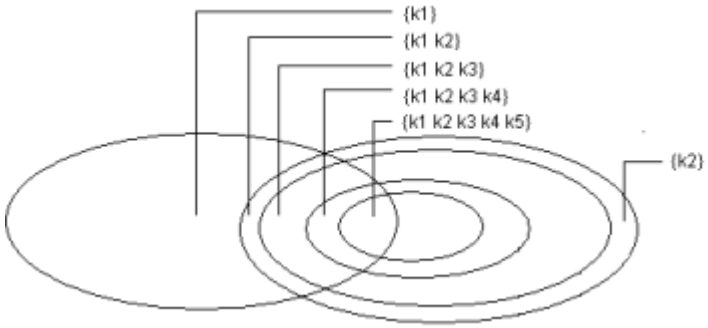

Figure 4. Greedy binary partition of {k1} and its subsets.

Repeat steps 1 and 2 using parent rows 2 through 4. For instance for parent row 2, k1 = loan and you would end applying a binary partition to discover the associated largest answer sets.

Optional: Once all five families have been identified, prune these by applying once a suitable ***If-Then*** condition.

It is also possible to identify strong association rules from a given family. Note from the example that the association rule with the largest confidence value was

{a e c} → {b}

that is,

{mortgage rates refinance} → {loan}

with a confidence value of

conf({a e c b}/{a e c}) = 59600/150000 = 0.3973 or almost 40%.

This suggests a strong association between these term sequences. Almost 40% of the documents mentioning the sequence *mortgage rates refinance* tend to mention next *loan*. The rule doesn't seem to occur by chance.

Last update: August 23, 2007

First published: August 8, 2007