Problem 1:

Table 5.1 can be viewed as a horizontal listings. Each column represents different terms. Cells in a row is the comparison of the three terms on a condition such as unfiltered, no numbers. Also without taking a look at the title, this table also can be an vertical listings, with rows representing different terms and columns being different entities of the term. The unusual part of this table is that for each column it has three subcolums.

Also, ratio\_is\_number is high for lexical features.

Queries can be “information retrieval preprocessing” or “preprocessing terms nonpositional postings tokens”. When using these queries, set of all relations are extracted from corpus, then perform a feature rank. The feature can be hits on header or the number of rows or hits on the leftmost column. Then do schema rank which is to test the coherence of the schema.

The false positive can be other preprocessing methods in information retrieval. This will reduce the precision.

Table 5-3 can be an vertical listings. Each row represents a term with the value of encoding and posting list. It can also be viewed as a enumeration which lists the encoding and postings list for “the”, “computer”, “arachnocentric”, respectively. Cells in this table has large value for ratio\_in\_number.

We can use queries such as “index compression encoding gaps” or “encoding gaps encoding postings list”, “encoding gaps arachnocentric” When using these queries, set of all relations matching these keywords are extracted from corpus, then perform a feature rank. The feature can be hits on title (encoding gaps) or on the leftmost column (arachnocentric). Then do schema rank which is to test the coherence of the schema. The false positive can be other preprocessing methods in information retrieval. This will reduce the precision.

When tables with more rows and more column, which ranked high on the list, are extracted but not close to topic. The expected table may be overwhelmed. Try to add the leftmost column fields to the query may avoid this problem.

Figure 9.8 is an Enumeration, in which the first column lists several words and the second column is their nearest neighbors. This figure can also be vertical listings.

The dist\_string in lexical feature is rather high for this figure, which indicates it is vertical listings to a large extent.

We can use “word nearest neighbors”. When using these queries, all relations matching this pattern are extracted, then perform a feature rank. The feature can be header match or the number of nearest neighbors found. Then do schema rank which is to test the coherence of the schema. The false positive in this scenario can be that the extracted neighbors are not close to the word semantically. In this case, an additional dictionary may be used to filter out those neighbors.

Figure 12.3 is a horizontal listings, with each column listing two distinct model. It can also be Attribute/Value, since the sub-columns in each cell look like attribute/value pairs. From the structure of this table, cell may contains characters more than 100 characters which violates authors’ assumption in the paper. The semantics is not clear for each cell, though we know this is a comparison between two models. So it is hard to extract this table because no good query can match this table very well.

The query is “comparison unigram language models”. When using this query, all relations matching the comparison are extracted, then perform a feature rank. The feature can be the number of hits on the leftmost column. Then do schema rank which is to test the coherence of the schema. The false positive in this scenario is that a lot of comparisons of two unigram language models are extracted but the expected result is buried into the massive tables. Additional keyword including words listed in the table may reduce the false positive.

Table 14.3 is horizontal listings which compare two cases that are with and without preprocessing of training set. The columns are layout vertically instead of horizontally.

The query can be “kNN preprocessing of training set”, “kNN classification training testing”. When using these queries, all relations matching “kNN” and “classification” or “preprocessing of training set” are extracted, then perform a feature rank. The feature can be the number of hits on the leftmost column. Then do schema rank which is to test the coherence of the schema. The false positive can be kNN classification with other formulas are extracted. This may be avoided by specifying the formula name.

Table 16.1 is vertical listings. Each row represents a distinct application and its entities are “what

is clustered”, benefit and example. The dist\_string in lexical feature is high for this figure.

The difficulty is how the query can be matched to this table since it is a table contains general information. The query can be “information retrieval clustering application” or “clustering application comparison”. These queries will return massive tables about the comparison. Ranking may be performed by hits of application names on the leftmost columns or the headers to improve precision.

Table 16.3 is vertical listings in which each row contains the docID and its text. It can also be viewed as enumeration which lists a series of documents. The table is supposed to have two columns instead of four. For easy layout, it is split into four columns. Also ratio\_in\_number here is roughly 50%.

It is hard to come up with the query because of the predicate is unknown. That is, we don’t know the relation between docID and document text. By looking at the header, the query could be “EM clustering parameter values” but this hardly can return this table, as it may return results that are completely no close. the better way to get this table is to do the header matching to rank tables. Another way is to do feature ranking, since ratio\_in\_number in this table is about 50%. We can find all tables that are close to this ratio.