COMP3430 Project Record Linkage
u6251843, Sunday October 21

Overview

The report consists of four sections, namely **Blocking**, **Linkage Quality**, **Optimal Settings** and **Data Quality**, which concentrates on blocking methods, classification and comparison parameter modification, optimal condition and data quality assessment.

Blocking

We introduce blocking in record linkage since we hope to remove obvious non-matching candidate record pairs. However, if the two matching records are classified into two different groups, two records will never have a chance to be linked. Suppose in two data sets, we have 5000 true matches, if only 4000 true matches pass through a blocking process, we at most obtain 4000 matches in the result even if the comparison function is perfect. This indicates the blocking process will impact on the number of remaining true matches.

(i) **Blocking key method and blocking key**

The method for blocking is Soundex phonetic encoding and the attributes chosen are *last_name* and *gender* in my choice. The value of pc is 0.837 for certain blocking, which means 83.7% true matches still 'stay' in the candidate record pairs. This is the optimal outcome I find after testing three main blocking methods, namely simpleBlocking, phoneticBlocking and slkBlocking.

(ii) **Blocking key criteria**

1. High attribute data quality

The blocking keys should not have missing values. If a key has a missing value, then as a result many records will be added into a block where the blocking key is empty, which makes it hard to decide whether these records are similar to each other or not. At the same time, the data in the block key (attributes) should be as clean as possible because dirty values make matching records classified into the wrong groups. Hence, in the provided data sets, *middle_name*, *street_address*, *phone* and *email* are not considered as key due to incompleteness while *postcode* and *birth_data* are not considered either due to too much dirty data in the attributes.

2. Uniform attribute values frequency distribution

The frequency distribution of values will influence the number of candidate record pairs that are generated. It is of advantage to select attributes which the values have uniform distribution as blocking keys to avoid too many records being inserted to the same block. Hence, *state* cannot be used as blocking key because its attribute values' skewness is around 0.71, which is far from uniform distribution.

(iii) **Trade-off between performance and linkage quality**

There is indeed a trade-off between reduction ratio (performance) and pairs completeness. Utilizing a blocking method that has a lower reduction ratio means a smaller number of candidate pairs is removed during blocking. This often leads to a higher pairs completeness. In the provided data set, I find when I reduce the number of attributes in the blocking key, the pairs completeness increases from 0.697 to 0.837 while reduction ratio decreases from 1.0 to 0.999, which indicates reduction ratio will not change as rapid as pairs completeness. Thus, I think it is acceptable to sacrifice some comparison space for a higher pairs completeness if the result reduction ratio is closed to 1.

The trade-off will change if the data quality is different. High-quality data will make blocking classify more true match record pairs into the same group while less non-matching pairs are classified into that group. If the data quality is perfect, we can use simpleBlocking directly and choose all the *names*, *gender* and *birth_date* as the key. All the true matches will almost be in the same group.

Linkage Quality

The classification algorithm, similarity threshold and comparison functions are three main variables which are related to quality of record linkage.

(i) **Classification algorithm and similarity threshold**

I start from classification algorithm, trying to find the best similarity threshold for each algorithm. The two graphs below represent the F-measure change in two different classification algorithms.

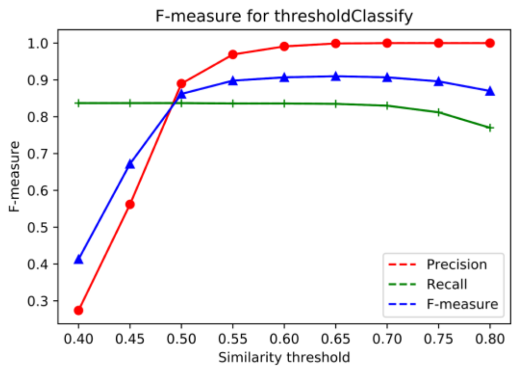

Figure 1

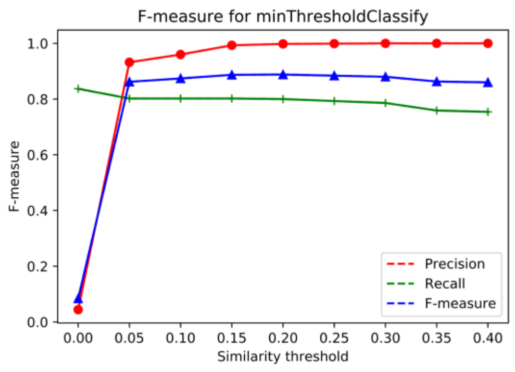

Figure 2

From the graphs, I observe linkage quality is particularly sensitive when similarity threshold is low (similarity threshold <= 0.5 in Figure 1 and similarity threshold <= 0.05 in Figure 2), but when the similarity threshold is greater than the border, linkage quality doesn't improve dramatically. Additionally, I conclude that the thresholdClassify algorithm works better than minThresholdClassify since some of F-measure in thresholdClassify algorithm reach 0.9 but all of F-measure in minThresholdClassify is below 0.9. At the same time, I detect the optimal threshold for thresholdClassfy, which is about 0.65. The value is used in weightedSimilarityClassify algorithm as well.

(ii) **Weight vector**

For weightedSimilarityClassify algorithm, I try to find the best weight vector so as to improve the record linkage quality. I start with sample vector (provided in Peter's code, vector is [2.0, 1.0, 2.0, 2.0, 2.0, 1.0]), increase 0.5, 1 and decrease 0.5, 1 for each attribute each. Since the change is really minor, I record all the true positive, false positive and F-measure in the table.

	-1	-0.5	0	+0.5	+1
First Name	4178, 24, 0.908	4179, 7, 0.910	4178, 5, 0.910	4177, 8, 0.910	4174, 11, 0.909
Middle Name	4178, 47, 0.906	4179, 16, 0.909		4176, 1, 0.910	4170, 1, 0.909
Last Name	4176, 1, 0.910	4176, 2, 0.910		4179, 9, 0.910	4180, 14, 0.909

Address	4177, 7, 0.910	4178, 5, 0.910		4178, 5, 0.910	4176, 5, 0.910
Suburb	4176, 16, 0.909	4177, 10, 0.909		4179, 4, 0.910	4180, 7, 0.910
State	4174, 1, 0.910	4176, 2, 0.910		4180, 11, 0.910	4180, 17, 0.910

Table 1: TP, FP and F-measure after minor adjustment

My goal is to increase the TP but reduce FP, but the table tells me the TP and FP nearly increases or decreases together. Hence, the best solution here is to attempt to find a vector which TP increases quickly while FP increases slowly. From the table, I observe increasing the weight of *last name*, *suburb* or *state* and decreasing other attributes will lead to a higher TP, so the optimal vector should have a higher weight on *last name* and *state*. Finally, the best vector found is [2.0, 2.0, 2.4, 2.0, 2.0, 1.3].

(iii) **Comparison function**
1. Attribute: Address -> Bag Distance

The edit distance is a good choice for address, but it has quadratic complexity, i.e. len(s1) * len(s2). However, the address is a long string, so comparing two address will cost too much time. The bad distance is a fast approximation of edit distance, which can handle the long string comparison efficiently, so I choose Bag Distance.
2. Attribute: Suburb -> Edit Distance

The suburb is a short string when it's compared to address. Hence, using edit distance to compare suburbs will not generate long time computation but provide an accurate comparison result.
3. Attribute: State -> Exact Comparison

The number of states is limited and deterministic, so I use the exact comparison to compare states in candidate record pairs directly.

Optimal Settings

(i) **Best linkage result**
The best linkage quality I achieve is:
1. True matching records found (TP) is 4177,
2. Non-matching records mistakenly judged as matching records (FP) is 1.

1. Since the best pairs completeness I find is 0.837, so the actual true matching pairs remains after blocking is 0.837 * 5000, which is 4185. The true matching records found by my program is 4177, which means only 8 true matching pairs lost during the comparison. The percentage of found pairs is $\frac{4177}{4185} = 99.8\%$ if we don't include the true matching pairs lost during blocking.
2. The decision tree classifier gives a 4182 TP and 0 FP, the best linkage result I find is very close to this optimal result.

(ii) **Evaluation metrics**
Accuracy -> 1.000, Precision -> 1.000, Recall -> 0.835, F-measure -> 0.910

From the previous table, I know the TP and FP increase or decrease together. When TP and FP increase, precision will decrease while recall increase. There exists trade-off between precision and recall. In my record linkage result, I try to reduce the number of FP as much as possible, so precision is very high while recall is a little bit low, but the F-measure is the highest one in all the conditions I attempt.

Data Quality

(ii) **Datasets comparison**

The data sets in this project seem dirtier than those experimented on labs. The table below lists the metrics for three typical data sets in the same model.

	clean-A\B-10000	little-dirty-A\B-10000	dataset-A\B
Pairs completeness	1	0.867	0.837
Precision	0.998	1	1
Recall	1	0.867	0.835
F-measure	0.999	0.928	0.910

Table 2: Metric comparison between data sets

From the table, I find when using the same parameter for three data sets, the clean data sets achieve the highest F-measure, followed by little-dirty one. The dataset-A\B has the worst result. I guess the data sets in project are dirtier, so both blocking and comparison function doesn't work as well as in those data sets provided in the lab.

(ii) **Assessment of data quality**

1.Completeness

Count the number of missing values for each attribute by python.
Formula: (all records – missing value records) / all records

2. Consistency

Find the maximum used format for each attribute and count the number of records with this format by pattern matching.
Formula: records with the same format / all records.

3. Uniqueness

Use blocking and comparison function to classify the similar records and remove the duplicate records.
Formula: Unique records / all records

4.Validity

Compare the data with metadata, if data doesn't conform to the syntax, then the data is classified into invalid value.
Formula: valid records / all records

5. Accuracy and Timeliness are hard to assess in the provided data sets.