Comp Sci 839 - Stage 3

Somya Arora, Mohammed Danish Shaikh, Swati Mishra (Group 16)

Performing Entity Matching using CloudMatcher

I. Table Schemas

The aim of this project stage is to perform entity matching on IMDb and TMDb tables created by scraping web data in project stage two. Both the tables have the following schemas:

ID	Name	Year	Duration	Genre	Actors

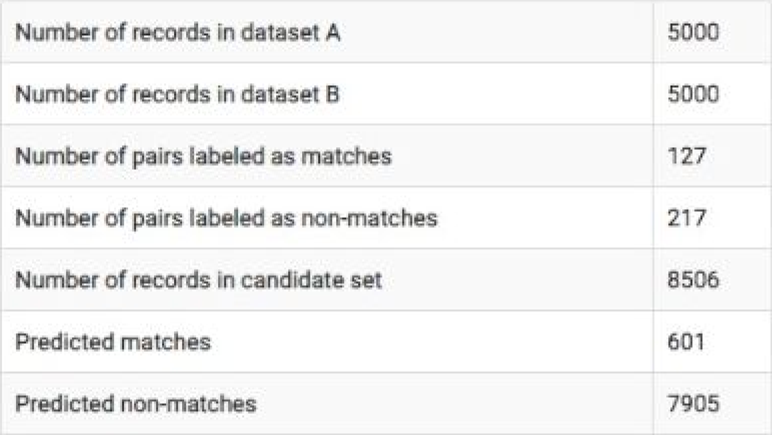
We'll be using the CloudMatcher system for matching entities between two tables. Our definition of match is "*Two movies X and Y match if they're produced in the same year and have the same name (the names can differ syntactically, i.e. X.name = Incredibles 2 and Y.name = Incredibles II is allowed)*".

II. Entity Matching Process using CloudMatcher

The high level process of performing entity matching using CloudMatcher is as follows:

1. Upload the IMDb and TMDb tables to the system, sample them to verify the upload has been done successfully.
2. Add ID (metadata) field to both these tables.
3. Perform active learning to help the system learn blocking rules. The system then applies these rules to the input tables to get the candidate set C.
4. Perform active learning on the candidate set C to help the system learn the matcher M that identifies matches.
5. The system finally gives the list of predicted matches P.

III. Experimental Data



Number of records in dataset A	5000
Number of records in dataset B	5000
Number of pairs labeled as matches	127
Number of pairs labeled as non-matches	217
Number of records in candidate set	8506
Predicted matches	601
Predicted non-matches	7905

IV. Process for calculating Precision and Recall

Our candidate set C had 8506 (> 500) pairs, hence we had to calculate the precision and recall using the following steps:

1. The density of random sample of 50 pairs from C was found to be 3/50.
2. We wrote a blocking rule to reduce the candidate set C to C’ containing 805 tuples. The candidate set C’ was obtained by applying the rule *IMDb.year* = *TMDb.year* (same movies should have the same production year). We debugged the rule using “debug_blocker” module to ensure that we aren’t dropping any true matches. The density of random sample of 50 pairs from C’ was found to be 44/50.
3. Since the density of random sample of 50 pairs from C’ was greater than 0.2, we sampled 300 additional random candidate pairs from C’. This is because we already had 100 (50 + 50) randomly sampled labeled tuples from the first two iterations (i.e. before blocking and after blocking). We labeled the 300 additional randomly sampled pairs to created a set of 400 labeled pairs.
4. We used the “estimate_precision_recall” module to estimate the precision and recall.

V. Precision and Recall of the Matcher

The bounds for precision and recall calculated using the “estimate_precision_recall” are [0.9886539264149898, 1.0025355008977854] and [1.0, 1.0] respectively.

Note: All code and data related to this project stage can be found here.