COMP20008 Assignment 2: Question 1C

Linkage Without Blocking

Distance Measurement

In task 1A my record linkage algorithm works by generating a score for each combination of Amazon and Google records based on the similarity of their titles, manufacturers (if present,) and prices. Although I originally thought a sequence-based text distance algorithm like `textdistance.ratcliff_obershelp` would be most effective at capturing the similarity between product titles (since I expected many similar substrings) I ended up using the `textdistance.overlap` method, as this token based algorithm was much more effective at eliminating false positive matches. I believe the other algorithms I tested were less effective as many of the products were of a similar category (software packages) and hence had lots of similar substrings regardless of whether they were a true match.

To compare manufacturers I used the edit-distance based `jaro_winkler` algorithm as I believed the recorded manufacturers of true positive matches would have a very low edit distance, and hence achieve a high similarity score with this method.

Finally, the discrepancy between the prices of the items was taken as a ratio of the price difference to the value of the item so that two sets of items would receive an equal deduction in score for any percentage change to their price difference.

Match Selection

Once these three distance weights were calculated, product pairs were determined to be a match if a function of their distances was above a defined threshold. The match's score was defined as:

$$match\_score = 3.9 \times (title\_similarity)^{1.5} + 0.05 \times (manufacturer\_similarity) - 0.35 \times |\frac{price\_difference}{amazon\_price}|$$

Where the similarities were values between 0 and 1, with 1 indicating a perfect match. The coefficients and exponents of each value in the *match_score* function were determined by running the record linkage algorithm on a training dataset and refined to optimise the sum of precision and recall on test data, it was decided to exponentiate the *title_similarity* value as this would amplify the effect of a near-perfect score much more than a linear weighting. Finally, pairs with a match score greater than 1 were considered to be a match; if two matches occurred for a single Amazon record, only the one with the greatest score was recorded.

This algorithm achieved $precision = 0.92$ and $recall = 0.93$ which could be further improved by optimisation of the coefficients used in the *match_score* function and perhaps selection of even better text distance algorithms.

Blocking

My blocking method works by generating blocking keys based on words that appear frequently in the record titles. Firstly all the strings from the Google titles are extracted and concatenated. This long string is split into a list by spaces and non-alphanumeric strings are removed. Next stop words are removed from the list by matching with `nltk`'s `stopwords.words()`. Blocks with keys of the words still contained in this list are then populated with amazon and google records as long as the record title contains the block key. Finally, blocks that contain more than 200 google items (as this is the larger of the datasets) are removed entirely. This removes keys such as `software` which matches with approximately 1000 google records and provides very little valuable information for linkage as a result.

Evaluation

This blocking method performs reasonably well, with a *pair completeness* of 80.38% and a *reduction ratio* of 93.60%. These are calculated based on the true positive, false positive, and false negative counts of 1,045, 280,475 and 255 respectively. The method may be further improved by using other data available about the items such as their price to link the records and capture some of the false negatives that slipped past my blocking algorithm. The blocking keys extracted from the titles may also be refined and further reduced to enhance the performance by reducing the amount of false positives.