Task 1 report

For task 1a, I made two lists (abt and buy respectively) where each list contains the name of product and its corresponding ID. My preprocessing is to split the text (name of product) into individual words, get rid of any punctuations in each word and make every word into lower case. Then, I extract all the suffix of name from both lists into another list called “suffix”. My definition of suffix is: if last second string of the name is “-”, then the last word of name will be treated as suffix; otherwise I assume the name does not have suffix. After getting all suffix, I loop both abt and buy lists, and then any name contains the suffix in both abt and buy list will be treated as actual matching pairs. After that, I remove all the names which were already matched from both lists and compare the similarity of the remaining names (i.e. any word in name 1 appears in name 2, then the similarity adds 1; if a word of name 1 does not appears in name 2, then similarity does not change) until all the individual words in each name has compared. So, the possible matching pairs is containing the highest similarity value among all the names and threshold is set to 1 (i.e. at least 1 word appeared in both name). The reason of setting threshold to 1 is because this threshold gives the least false positive results. After removing the duplicated matching pairs, those remaining possible matching pairs would be treated as the actual matching pairs. The actual matching pairs from both suffix matching and similarity comparison are the final output of 1a.

For task 1b, the preprocessing (get rid of punctuation) method, suffix extracting and matching method, method of removing matched names from both lists are exactly the same as 1a. The reason I was doing suffix matching in 1b is: the suffix of each matching pairs is unique, which can be used as block keys. Then, I realized the prefix of each name is mostly the brand of products, so I also used prefix as block keys, and the names having the same block key (prefix) are stored in the same inner list. The reason I was doing this is: the huge amount of data would cause the execution time very long, so the division of the list into many small lists by prefix blocks helps reducing the number of comparisons and increasing the accuracy. Then, if any pairs have the greatest similarity and greater than threshold (at least 1 word in both names are the same), it will be directly treated as the actual matching pairs because we need as more true positive pairs as possible in order to improve the pair completeness. (Noting this threshold is the most appropriate threshold because increase threshold would decrease the value of Pair Completeness; decrease the threshold to zero does not make sense for finding the similarity). These actual matching pairs are the output of 1b.

The result of 1a is: two columns of IDs: the left column is the idabt and right column is the idbuy. There are 149 rows in total where each row is my unique matching pair. As a result, my recall and precision are both 0.8523 where True Positive Count is 127, False Positive Count is 22, False Negative Count is 22, True Negative Count is 61581 respectively. It is an acceptable result but I think it still can be improved. I treat every word is the same, but actually, there could be some words are more important than other words. For example, the brand of product (suffix) could be more important than some other descriptive words such as color, size; equivalence of two consecutive words are more important than equivalence of random two words etc.

The result of 1b is: two csv files where the first csv has a column of blocks and a column of abt IDs, the second csv has exactly the same column of blocks (with same order) and a column of buy IDs. Every row in “abt_blocks” csv would match the corresponding row in “buy_blocks” csv as the two IDs are matching pairs. The outcome of task 1b is: 1509 rows in both csv; Pair Completeness is 0.9626; Reduction Ratio is 0.9765; Execution Time is 0.003805s; True Positive Count is 1056; False Positive Count is 26651; False Negative Count is 41; True Negative Count (Assuming unique blocking) is 1152704; Abt ids in block is 100; Buy ids in block is 100 and Comparisons required is 10000.
This result also could be improved. The price of each product is potentially useful in this Question. What I mean here is there could be some methods such as rounding the prices, which could be used to relate the similar price products, which could potentially improve the possibility of matching true positive pairs as well as reducing the number of false positive pairs.