Lizette Galarza
CS 4400X Final Project

Link to Github: https://github.com/lgp3/CS4400_Spring21_Project

- Submission CSV: solution/machine_learnt.csv
- src/ contains the source code to block, generate features, train, and test
- data/ contains the data provided for this project and intermediate CSVs

Data Knowledge

The first step to take was to understand the dataset. The size of the dataset means that there are 2555 * 22075 = 56,401,625. This huge number would have to be reduced to be able to handle the feature engineering, testing, and training on a laptop. Blocking could be performed to reduce the size of the dataset.

By looking into which pairs were matches, it was found that many of the true matches had either the same brand or a brand that was extremely similar (e.g. “tp link” and “tp-link”). This fact can be used to implement a blocking method that will capture most of the true matches.

Additionally, it was found that there was a lot of missing data. This would have to be accounted for during blocking, feature engineering, and training.

Blocking

A few different blocking methods were used.

First, simply blocking on the brand was used. While this did help to reduce the dataset to a manageable size, this was missing some true matches that did not have exactly identical brands. This led to the would lead the model to be less performant

To improve upon this, a method of fuzzy text matching was implemented. To do this, first all the brands from both tables were found. Then, the brands in the right table were replaced with the brand from the left table that was most similar. Most similar in this case is performed using the Levenshtein distance[1]. Specifically:

$$similarity(stringA, stringB) = \frac{\max\left(len(stringA), len(stringB) - leventshteinDistance(stringA, stringB)\right)}{\max\left(len(stringA), len(stringB)\right)}$$

Throughout the rest of this document, this will be referred to as similarity.

This similarity metric was used for blocking. If two brands had a high enough similarity, they are considered to be the same brand for blocking purposes.

Additionally, blocking on the category was performed. If two categories were extremely similar, they were considered the same for blocking. The results of the brand blocking and the category blocking were then combined.

This resulted in 348,490 possible pairs to build the dataset upon.

Feature Engineering

The features created were the following:

- Price Diff
  - The difference in prices of the two entities
- Model Similarity
  - The "similarity" between the model numbers
- Category Similarity
  - The "similarity" between categories
- Brand Similarity
  - The "similarity" between the brands
- Title Match
  - Using the "similarity" between the titles is more dubious. To determine how similar the two titles were, a more complicated method was invoked. To determine this similarity, each title was converted into a list of word vectors using Google's pre-trained Word2Vec model[2][3]. This list of vectors were compared by finding the average of the max cosine similarity between each word in each phrase. In pseudo code:

```
phrase_a = "some phrase"
phrase_b = "some other phrase"

vector_list_a = to_vector_list(phrase_a)
vector_list_b = to_vector_list(phrase_b)

best_match_scores = list()
for word_vec_a in vector_list_a:
      best_match = - 2
      for word_vec_b in vector_list_b:
            match = cosine_similarity(word_vec_a, word_vec_b)
      if match > best_match:
            best_match = match
      best_match_scores.append(best_match)

final_score = mean(best_match_scores)
```

With these features created against the data, a model could be trained and tested.

Model Selection

Multiple popular models were tested against the dataset to determine which should be used ultimately. The tested models included: Random Forest, SVM, Linear Discriminant Analysis, Quadratic Discriminant Analysis.

The models were tested by splitting the data into training and testing data. The Random Forest classifier consistently had the best f1 score, so it was selected to be used for the final classification.

Output

The Random Forest Classifier was then trained against the available training data. The trained model was then used to classify all the data that had passed blocking. The training truths were then removed, leaving the final predicted true matches.

Appendix

[1] Overview of Levenshtein Distance: http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm

[2] Word2Vec Paper: https://www.cambridge.org/core/journals/natural-language-engineering/article/word2vec/B84AE4446BD47F48847B4904F0B36E0B

[3] Google's Word2Vec Model: https://code.google.com/archive/p/word2vec/