CS 784 Project Stage 4 Report

Kritika Rai, krai2@wisc.edu
Xia Wan, wan5@wisc.edu
Zhen Zhang, zhen.zhang@wisc.edu

- Precision: Recall: F1: 96.10%; Recall: 83.20; F1: 0.8918

- Details for matcher M
  - Stage 3
    - Details
      We used Random Forest with 100 trees. For the features we extracted, we applied feature extractor to top 12 attributes. After feature selection,, we extracted:exact matching for “Product Segment”
        - edit distance and 3-gram and word based Jaccard and TF/IDF for “Product Name”,
        - exact matching for “Product Type”, 3-gram based Jaccard and TF/IDF for “Product Short Description”
        - edit distance and 3-gram and word based Jaccard and TF/IDF for “Product Category”
        - 3-gram and word based Jaccard and TF/IDF for “Brand Name”
        - exact matching with normalization for “Product Country Origin”
        - exact matching for “GTIN”
        - exact matching for “UPC”
        - exact matching for “Manufacturer”
        - exact matching for “Manufacturer Part Number”
        - exact matching for “Product Color”

      The threshold for “MATCH” and “MISMATCH” were set as 0.651 and 0.349, respectively.
    - Set Y: Precision: 96.25%; Recall: 82.78%; F1: 0.8900
      Blind data set: Precision: 96.7690192484; Recall: 84.4431113777; F1: 90.1868659904
  - Stage 4
    - Details
      Based on the matcher we built on stage 3, we applied handcrafted rules after the learned model. With looking into the wrong labelled data, we modified the threshold to get higher recall and lower precision and wrote some rules to increase the precision while keeping recall relatively high. The details for rules show as following (preprocessing the values of attributes to handle case-sensitive conditions):
        - For ink cartridges, if only one product contains “Color” attribute and the value of “Color” shows only in “Product Name” of the other product, the prediction for the product pair is “MATCH”

- If the products both have “Product Name”, “Product Segment”, “Product Type”, calculate Jaccard / (1.0 - levenshtein / max_length_product_name) for “Product Name”. If this score is greater than some threshold, the prediction is “MATCH”
- If the products both have “Product Name”, “Product Segment” with different values, “Product Type” with different values, the levenshtain of product name is greater than 5 and “cooler” and “cooling” show in the product name respectively, the prediction is “MATCH”
- If the products both have “Product Name”, “Product Long Description”, one product long description contains the product name of the other product and the product names show in some predefined dictionary, prediction is “MISMATCH”
- If “refurbished” and “remanufactured” shows only in one product name, the prediction is “MISMATCH”
- If the product names only differ on only one word (other words lie in the same order), the final prediction could be “MATCH”, “MISMATCH” or original prediction based on if the word shows in some dictionary
- If original prediction is “MATCH” but the values of “Assembled Product Length” are different, the final prediction is “MISMATCH”
- If original prediction is “MATCH” but the values of “Color” / “Actual Color” are different, the final prediction is “MISMATCH”
- The product names could contain important model and serial number and the have high similarity except the model and serial number, but there are actually “MISMATCH”. Model and serial number extraction rule combined with the comparison rule are used to refine the “MATCH” and “MISMATCH” results from RFC. The length of the possible model string, jaccard score, the information in long and short description are considered.

  - Precision: 96.10%; Recall: 83.20; F1: 0.8918

- Analysis
  - Why fails
  - In stage three, we found RFC is the most accurate algorithm among our tried machine learning algorithm. Thus, in stage four, we tried to add some post rules to refine the predicted results from the RFC. We finally add 7 working rules. However, we could not improve our Precision and Recall too much with our designed framework.
  - We want to build more general rule to improve our results. But, with a lot of time and trying, finally they are not working. For example, the model and serial number extraction rule could not fit in the variation situation of the production name in the

source data, which results in the recall decreased dramatically with the improvement of precision.

- From the second lesson, we notice that we should do some clean and feature extraction before apply the training and post rule pipeline. Without the pre-transformation, our designed RFC and post rule only catch a few samples pairs.
- The fourth lesson is that the importance of the dictionary method to introduce our domain knowledge. In our pipeline, we used the dictionary for colors to help find the false positive. Right now, we think we should add more dictionary to distinguish production, for example, ‘i3’ v.s. ‘i5’ for models.

○ Future work

Based on our lessons, in the future, we want to do following improvements:

- Build the information extraction processing to standardize the following features: brand, color, model, condition, size, length before RFC training and applying rules.
- Utilize our prior domain knowledge. In the pre-information extraction and rule apply step, more domain knowledge should be included with dictionary based method or rule based method.

- Discuss of py-stringmatching package

  We used both modules in the package. Most of them work well and are easy to use. In simfunctions, levenshtein returns integer 0 if the inputs are same and decimals for other conditions. The inconsistency will cause some trouble for some machine learning algorithms. We modified it to return 0.0 instead of 0.