CS 784 - Final Project Report

Group-05: Danish Khan, Mehreen Ali

Introduction

In this stage, we picked the problem of predicting the labels for unlabeled pairs of products by using a product matcher, M. Our labels include “Match”, “Mismatch” and “Unknown”. This classic problem of entity matching is hard to solve due to a variety of reasons. Firstly, it is prone to human errors like misspelled description in one product or a wrong description of another product entered manually in the source database. Secondly, there might be missing values to certain attributes which help in matching two products as same or not. Likewise, there are other factors which make this particular problem hard.

For this project, we were given a set of twenty thousand labeled product pairs, and we had to randomly split them into two sets X and Y, each having ten thousand pairs. We used X to develop a matcher M, then eventually applied M to Y and reported the accuracy of M on Y. We also applied M to a blind data set B provided to us on which we were evaluated in terms of the accuracy of M on B, to know how well M is as a product entity matcher.

We have used supervised learning as the basis of our approach. Looking at the product attributes, we observed that there are certain attributes which represent the product semantic better than others. To name a few, `Product Name`,`Product Short Description`, `Product Type`, `Product Segment`, `Brand`, `etc.` After such key attributes are identified, we use their corresponding attribute values to generate feature vectors using various similarity measures for each product pair. With these feature vectors generated for the training data set X, we use them to train our matcher (M) using Random Forests based machine learning technique. Finally we apply this matcher, M on the unseen dataset Y using 5-fold cross validation to compute and report Precision, Recall and F1 Score. Detailed description of our understanding of precision, recall and F1 score is given in Appendix-B. Following sections discuss about our approach and results in detail.

Entity Matcher Description

Our product entity matcher, M, follows a supervised learning approach. We extract a feature vector per product pair in given dataset. Features in this feature vector consists of different similarity score between important product attributes. These feature vectors are used then used to train our matcher which is eventually used to predict labels on our test data. Below are details about individual steps involved.

Feature Extraction

a. We focus on identifying the most important attributes. Once identified, these are the attributes for which the feature vectors would be generated. For this task, some of the attributes which captured the product semantics well were obvious choice for feature vectors. For instance, `Product Name`, `Product Type`, `Product`

`Segment, Brand, Product Short Description` etc. We have picked up most frequent attributes to minimize the error due to missing values.

b. Clean the input data for further processing. This includes processing selected attribute values in previous step. Specifically we made following changes to original data input.
   i. Convert the attribute value to lowercase characters. Also, Remove all consecutive spaces.
   ii. Remove all the less important words. This includes connectors, conjunctions, etc. For ex. “The”, “a”, “for”, etc.
   iii. Replace “.”, “-”, “/”, etc. characters with a space.
c. Find similarity scores as features. We hypothesized that having various types of similarity measures for each attribute would increase the overall accuracy because different similarity measure will capture different semantics and hence better overall semantics. Our similarity measures which include **Jaccard, Jaro, Levenshtein, needleman_wunsch, tf/idf and monge_elkan**. We used ***py-stringmatching*** package for similarity measurement task. Our experience with it is described in **Appendix-A**
d. Missing value imputation of certain attributes is done with the mean value of that attribute using imputer function of scikit library. Those attributes which can distinguishably classify a product may increase the overall accuracy, even if their missing value is not significantly low. For eg, brand name and actual color attributes.
e. Once we have preprocessed the attribute values, we use them to generate feature vectors using various similarity measures for each product pair.

Training Matcher using generated features:

a. With these feature vectors generated for the training data set X, we use them to train our matcher (M) using Random Forests based machine learning technique. Besides this, models like Decision Trees, Random Forests, Naive Bayes, Support Vector Machines, Logistic Regression and other ensemble approaches were also experimented with. After conducting several rounds of experiments, we observed that Random Forests worked best in all the scenarios, and hence we chose Random Forest.

Evaluation on tuning set:

a. Finally we apply this matcher, M on the unseen dataset Y using 5-fold cross validation to compute and report Precision, Recall and F1 Score (**Appendix-B**).

Five fold cross validation results for our entity matcher on given labelled dataset for Project Stage 3 (Part 1):

Mean Precision	0.967783745835
Mean Recall	0.915758620272
Mean F1	0.941049901576

Table 1: Best Results for stage 3 (Using **Random Forest**)

Results for our entity matcher on the blind dataset reported back to us:

Reported Precision	85.2073288332
Reported Recall	88.3423315337
Reported F1	86.7465148243

Table 2: Results on blind dataset (Using **Random Forest**)

Explanation for significant difference in precision in reported result:

After discussing our results with Prof. Anhai, we came to the conclusion that our results were different on the blind dataset from the given test dataset Y, due to overfitting problem. We did not take initially a tuning set while developing our matcher and used the test set for cross-validation. This exacerbated the overfitting problem and also led to learning some of the test examples. This caused our model to perform poorly on the blind data set. We have again modelled our matcher after correcting this problem and have improved it using a tuning set to get the minimum desired precision of above 96% on the test set Y from the given labelled dataset.

Further Improvements in the matcher

Here we explain all the modifications that we made in this stage:

1. We use a larger number of attributes (40) instead of simply relying on the ones that had low missing values. For instance, including attributes like “actual color”, “size” etc which don’t seem important led to small improvements in accuracy.
2. We tweaked the threshold of our random forest classifier to get a better accuracy. We choose 0.6 as the threshold based on our experimental observations.

3. Apart from using machine learning, we also added some Rules to decrease the number of false positives and false negatives. For instance, if the brand name of the two products do not match, we declare a mismatch immediately instead of using our learned model to make a prediction.
4. We added TF/IDF similarity measure to generate more features while learning our model. This similarity measure is particularly helpful in detecting false positive cases where the product description is almost the same except the model is different. For instance, Iphone-4 and Iphone-4s. In such cases, if we use a similarity measure like edit distance, we will get a high similarity score. However, if we use tf/idf measure, then we get lower similarity score as it looks for similarity within non-similar/non-frequent words.

Five fold cross validation results for our improved entity matcher on given labelled dataset for Project Stage 4:

Precision	0.965127202231
Recall	0.90684457907
F1	0.935078594879

Table 3: Best Results for stage 4 (Using **Random Forest**)

Discussion

We learnt in class that there are lots of factors that may affect the value of Precision and Recall. We have described the some of the possible factors in this section.

1. Subtle differences in the product names are not caught by the matcher and this causes the products not to match. Like desktops or laptops have same brand name, but, have different configurations. Rule based classifier having extensive list of rules to capture all such scenarios can help reduce this problem.
2. Missing attribute values : There were a lot of missing values for some important attributes like brand, category, color etc. To handle this, we plugged in missing values for each of the attributes with the mean value of that attribute over the training set and used that to learn our model. The mean value is a good approximation for general cases, however it does not capture corner cases perfectly. Here again, we can reduce this problem by adding rules which can fill in missing values based on domain knowledge of other attributes.

3. False positive and false negative trade-off: It is not possible to simultaneously reduce both false positives and false negatives to 0. For instance, when we replaced **edit-distance** with **tf/idf** measure while generating feature for **"Product Short Description"** and **"Product Brand Name"** to detect false positives like **Apple** and **Apple, Inc**., we also resulted in several false negatives as it lead to lower similarity score in some genuine matching cases. We sticked with similarity measures which resulted in highest possible accuracy.
4. Consider products like 'HP 26 Black Original Ink Cartridge' and 'HP 20 Large Black Original Ink Cartridge', now again these products are different slightly and it is difficult to write rules to capture these changes in product names.

Appendix - A: *Discussion on py-stringmatching package* (Extra Credits)

a. The features we wrote primarily used the package and helped us abstracting all of similarity calculation.
b. We used **Jaccard, Jaro, Levenshtein, needleman_wunsch, tf/idf and monge_elkan**. We used both the delimiter approach and qgram approach. The package is very helpful and easy to use.
c. We found the matcher took time more time with more complex measures which is expected, but there is a possibility of improving the implementation technique. May be multithreading, caching, etc.
d. APIs handle null values while calculating similarity scores. For example, similarity score between a given string value & null string does not give Null pointer exceptin. For example, for "Sony" and NULL, the package will return 0 for Jaro and 4 for Edit Distance.
e. Intuitive, user friendly API's for various similarity measures. One stop shop for all major similarity measures. Can be extended further to include more advanced measures like Tversky index, etc.

Appendix - B

Precision and Recall

We used following method to calculate precision and recall for our brand extractor function on test data.

Precision = (#True_Positives)/(#True_Positives + #False_Positives)

Recall = (#True_Positives)/(#True_Positives + #False_Negatives)

True_Positive: Count of products where brand extracted by function is same as brand assigned manually on sample data set.

True_Negative: Count of products where brand extracted by function is wrong from sample data.

Fase_Positive: Count of products where brand extractor function predicted a brand but actual sample data didn't have a brand name.

False_Negative: Count of products where brand extractor function didn't predict a brand but actual sample data had an associated brand.

Appendix - C

"product name" "product type" "product segment" "brand" "gtin" "upc" "category" "product short description" "color" "size" "actual color" "country of origin: components" "manufacturer part number" "manufacturer" "assembled product length" "assembled product width"	"assembled product height" "warranty length" "condition" "composite wood code" "E-waste-recycling compliance required" "type" "cpsc-regulated indicator" "has Mercury" "operating system" "multipack indicator" "battery type" "screen size" "features" "number of batteries" "hard drive capacity" "processor type"	"energy star" "ram memory" "memory capacity" "processor speed" "connector type" "processor core type" "warranty provider"