**Data Analytics Term Project Status Report**

Group: Katherine Rodgers and John Merranko

**Introduction**

The purpose of this project is to predict the relevance of search results on HomeDepot.com (as compiled for the 1/18/2016 Kaggle competition) making use of NoSQL data processing and advanced text mining techniques. The data consist of search terms (strings of text) and products resulting from each search. Product data include product names, descriptions, and for some products, a collection of attributes (all strings of text). In the training dataset, raters have rated each pair of searches and resulting products on a three-level relevance scale (1 = irrelevant, 2 = partially or somewhat relevant, 3 = perfect match), and the task is to train a model using these data to try and predict the relevance of search and resulting product pairs in the test dataset. Model performance will be evaluated on the Root Mean Squared Error (RMSE) defined below:

**Data Processing**

We have implemented the initial data processing using Python. The data within the provided product\_descriptions.csv file consists of a product\_uid and description for each product. The train.csv file includes an id, product\_uid, product\_title, search\_term, and relevance value for each search. The attributes.csv file contains differing amounts of additional information for a subset of the products and consists of a product\_uid, name, and value in each line of text. The name refers to an attribute’s name and the value is the value for that specific attribute name. The first part of processing the data was to convert the comma separated files into a more easily separated format for Python due to the many commas within the product description and attribute data. This was done using R to initially read in the csv files and then write them using a pipe separated text file format. The Python script reads in the pipe separated train, attributes, and product description files where each line undergoes a process of string manipulation. We utilize the Natural Language ToolKit, NLTK, to remove non-essential stopwords for each line. We also remove some punctuation within each line of text to get only a series of words. Due to some of the typos within the provided data files and also the process of removing punctuation, issues of camel-case scenarios and letter-digit combinations are also accounted for. Due to the fact that any given product could have any number of attributes and the name of an attribute is non-uniform among the products, product data can be considered highly unstructured. Because of this, we decided to use the NoSQL database MongoDB to store a collection of product documents. The following is an example of a product document output in JSON format:

{

"attributes": [{

"name": "Product Width in",

"value": "3"

}, {

"name": "Product Weight lb",

"value": "0.26"

}, {

"name": "Product Height in",

"value": "3"

}, {

"name": "Product Depth in",

"value": "1.5"

}, {

"name": "Number Pieces",

"value": "1"

}, {

"name": "MFG Brand Name",

"value": "Simpson Strong-Tie"

}, {

"name": "Material",

"value": "Galvanized Steel"

}, {

"name": "Gauge",

"value": "12"

}],

"product\_description": "Not angles make joints stronger also provide consistent straight corners Simpson Strong-Tie offers wide variety angles various sizes thicknesses handle light-duty jobs projects structural connection needed Some bent skewed match project For outdoor projects moisture present use ZMAX zinc-coated connectors provide extra resistance corrosion look Z end model number Versatile connector various 90 connections home repair projects Stronger angled nailing screw fastening alone Help ensure joints consistently straight strong Dimensions 3 in x 3 in x 1-1/2 in Made 12-Gauge steel Galvanized extra corrosion resistance Install 10 d common nails #9 x 1-1/2 in Strong-Drive SD screws",

"product\_details": "Install 10 d common nails #9 x 1-1/2 in Strong-Drive SD screws Galvanized extra corrosion resistance Made 12-Gauge steel Dimensions 3 in x 3 in x 1-1/2 in Help ensure joints consistently straight strong Stronger angled nailing screw fastening alone Versatile connector various 90\u00b0 connections home repair projects",

"product\_uid": 100001,

"searches": [{

"product\_title": "Simpson Strong-Tie 12-Gauge Angle",

"relevance": "2.5",

"search\_id": "3",

"search\_term": "l bracket"

}, {

"product\_title": "Simpson Strong-Tie 12-Gauge Angle",

"relevance": "3",

"search\_id": "2",

"search\_term": "angle bracket"

}]

}

For ease of searching, we decided to concatenate all attribute values for a given product\_uid that uses “bullet” within its name (such as “Bullet01”, “Bullet05”) into one category, product\_details. We also used “name” and “value” as keys for lookup as opposed to the name as the key and the value as the value. We store it that way because of the non-uniformity of the names themselves. We can search through the attributes by seeing if the “key” is relevant from the given search term and then getting the associated “value” for word matching. Now that we have implemented a way to store the information for each product, we will then access the database in R via the MongoDB library function.

**Proposed Methods for Data Modeling**

We have designed the model we would like to use but have not fully implemented it in R yet. Our design to tackle this problem will utilize a methodology similar to the term frequency-inverse document frequency (tf-idf) calculation, which seeks to model how important a word is to a document in a collection. In our project, we propose to calculate the proportion of words (or measures) in the product name, description, and attributes (separately) that match text in the search term, and use these proportions as predictors in a three-level classifier model in order to predict relevance between search term and product pairs. Theoretically, if a large proportion of words in the product’s name, description, and/or attributes match words in the search term, the search result should be quite relevant. Conversely, if a small proportion of words in the product’s name, description, and/or attributes match words in the search term, the search result should be quite irrelevant.

We propose training a three-level classifier model with predictors that measure the word-match proportions described above and testing this model on the provided test sample. Below are basic mathematical expressions of our proposed methods:

, *i*=1,…,n and *j*=1, 2, 3; where:

= predicted relevance score (scale from 1-3)

– represents three-level classifier technique (e.g., random forest, multinomial logistic regression, etc.) found to have lowest RMSE when predicting as a function of predictors , defined as:

*=*

*=*

*=*

Note that before computing predictor variables, nuisances such as stopwords, (e.g., of, and, the, etc.) and punctuation (with the exception of punctuation joining compound words or numeric expressions) will be removed, and strings will be cleaned to remedy issues in the data such as the unintended splicing together of words either already in the data or after punctuation removal. Predictor variables may also be log-transformed to improve model fit. It is also important to note that not all products have product attributes. Thus, the variable will not be calculable for all search term and product pairs. To remedy this, values may be imputed (implementing either simple techniques such as median-imputation or more complex methods such as multivariate imputation by chained equations), or else we can instead implement classifier techniques that can internally handle missing predictor values (e.g., random forest). Either way, the predictors will be used to train a model estimating the probability that a given search term will have perfect, partial, or no relevance to the resulting product. As previously mentioned, model performance on the test sample will be evaluated using the Root Mean Squared Error.

**Proposed Software Usage**

We intend to use Python’s *PyMongo* to create a collection of product documents inMongoDB. The majority of data manipulation occurs in this stage before it is inserted into the database. We intend to perform model building, training, and testing in R.