Project on Home Depot Product Search Relevance

Ja-Jan Hsu (jah247), Leilei Liu (lel74), Zhaoyan Ai (zha4)

I. Introduction

Every merchant is willing to know the exact goods shoppers want to buy by analyzing shoppers' basic description of the goods. So the Home Depot product search relevance model can be used for most of the online shopping websites.

The final goal for the competition is to create a model which can predict the relevance between a list of goods and the exact item in shopper's mind by giving each good a value from 1 to 3. If the value is 3, it means the item is highly relevant in this model.

According to the requirement of the competition, the prediction model should be speed, accuracy and delivering a frictionless customer experience. How to design the prediction model, how to choose the impact factors and how to set the value of each impact factor can be the most significant part of work in this project.
Data files

In this competition, we have five data files and one relevance_instruction word file.

Train.csv file: there are five data fields(id, product_title, product_uid,search_term and relevance).
Test.csv file: there are four data fields(id, product_uid, product_title and search_term).
Product_descriptions.csv file: there are two fields(product_uid and product_description).

Attributes.csv file: there are three field(product_uid, name and value).
Sample_submission.csv file: there are two data fields(id and relevance)

From the sample_submission.csv file, we can notice that we'll need to predict a relevance for each items. The resource data we have are train.csv, product_description.csv and attribute.csv files. We'll need to find out the relations between each data field and predict a most accurate relevance for each of them. From the relevance_instructions file, it specifies how relevances are being rated.

II. What we have done?

1. Clean-up data

Data clean-up is a process of correcting inaccurate, redundant, or corrupt data and to make data in a consistent format. We converted all the characters to lowercase, remove special characters,and conducted misspell check. We used Python built-in method to convert characters to lowercase and remove/replace special characters. For example: We removed ",", "$", "Ã¥Âi", "+",";", "&""&amp","?", "-", "#" ,"(" ,")" and replaced "//" to "/", ".." to "." / " to " ovr " etc. We also replaced some other characters so that the words would be more self-descriptive. For example: We replaced " lb " to " 1lb ", replaced " sq." to " 1sq ", replace" sq " to " 1sq ", and replace " v. " to " 1volt ". We used inflect package to convert all number words to integers(p.number_to_words()). We tried to remove stop words but we found the result became slightly worse.

Lastly, we used two ways to conduct misspelling correction. We first used JAVA's jSpellCorrect library and fed the algorithm with a training data that we created to correct words in our Home Depot data. The accuracy became better; however, we found our training data was not thorough and cannot capture all the spelling errors. Thus, we referenced the method mentioned in the scripts thread "Fixing

Typos" in Kaggle and use the spelling check dictionary that created by the author to implement the spelling check. After trying these two spelling check experiments, the overall accuracy has greatly improved.

2. Stem data

Stemming data is a process for reducing inflected or sometimes derived words to their word stem, base or root form, generally a written word form. In this project, we rewrote stemmer based on SnowballStemmer from nltk.stem package. The new stemmer was customized only for Home Depot dataset to improve the accuracy of the prediction.

3. Feature selection

As our primary goal of this project is to predict a numeric value 'relevance' based on different text information, it is therefore important to select features wisely after the raw data is pre-processed. Based on some insightful scripts on Kaggle.com, our project first adopted a straightforward yet effective approach, counting words that appear in both search item and product title and words that appear in both search item and product description. These two features were fed to a random forest regressor to train the model and our initial result was promising. Such success inspired us to delve into the problem from a semantic point of view and ngram naturally became our point of interest.

Our first approach using ngram model was word-wise, which is counting the number of two consecutive words and three consecutive words that appear in both search item and product title and in both search item and product description. Our ranking was improved significantly.

We then moved on to further explore the power of ngram model. Rather than counting bigrams and trigrams from the point of view of words, we were curious

how the performance might be affected if we look at the problem at the fundamental level - characters. We then fed the number of two consecutive characters and three consecutive characters that appear in both search item and product title and in both search item and product description to the regressor model, and we obtained a better result.

III. Future Work

The Home Depot project was the first data analysis project in Python for all of us. Since we were not familiar with Python and the techniques in data analytics, we tried some basic methods, like clean-up data and stemming data. Normalizing data is one way we were willing to do but didn't spend much time to try on. Feature transformation is another regret for us. Feature transformations add background experience to the input data, enabling the machine learning model to benefit from this experience which may help us improve the accuracy of our prediction.