Ramin Tavassoli

Amazon Sentiment Analysis

Kaggle
CS – 565

Methods

In this project, we trained a model on a dataset from Amazon.com enumerated with 20+ features over 670,000+ records. The task was to predict the star ratings of starless entries of the test set given the known star ratings reflected in the training set. Data preprocessing was the main component of this project and the myriad ways of text mining and massaging the data provided for an exciting series of trials and errors.

Before looking at the data, I thought about choosing the most informative dimension using PCA analysis. But eyeballing through the data, it became swiftly apparent that the only feature worth analyzing was the column associated with reviews customers gave in response to their acquired applications. I purged my training set of all other columns accordingly. I then proceeded to identifying the right approach to preprocess the data. I stumbled upon an exciting package called “Pattern” which identifies grammatical structures within text. The module tags each word with a specific tag corresponding to its role in the sentence. For example, the sentence “what a great day” would associate tag = ‘RB’ with the word ‘great’ indicating its role as the adverb of the sentence. Tag = ‘NN’ is assigned to ‘day’ indicating that it is a noun. This package came immensely handy in processing data as it rendered the use of tokenization and ‘stop words’ removal obsolete. I appended any word that played the roles of noun, adjective or adverb into a list and replaced every record with a list of these words.

Then I deployed the “sentiwordnet” from “nltk.corpus” to quantify sentiments associated with the collected words. I used two parameters “{a}” and “{n}” corresponding to adjective and noun and compounded the positive scores by adding the sentiments associated when a word is used as an adjective or as a noun. For instance, the word ‘fun’ has two different sentiment scores when considered as a noun versus an adjective. Negative scores were also computed as such. I replaced every entry of the “review_text” column with corresponding scores. This ended the preprocessing step.

I deployed the kmeans++ clustering algorithm and distributed my data into 5 clusters without regards to the target star ratings at first. Only after data was clustered, I found the discrete probability distribution of star ratings among the 5 clusters. Therefore, each cluster became associated with a probability distribution. I then ran Kmeans++ on the test set and assigned each entry the probability distribution of the cluster they happen to fall into. For example, if training the model yielded (0.1, 0.2, 0.2, 0.1, 0.4) for the first cluster (assuming we have labeled clusters), then if a record in the test set falls in the first cluster, the aforementioned probability for star rating distribution would be assigned to that point.

In Summary, I used a combination of nltk and pattern modules to preprocess the data by quantifying their sentiments. I then applied Kmeans ++ clustering paradigm and associated a distinct probability distribution of star ratings to each of the clusters. I then used this information to assign pmfs to corresponding data entries from the test set.

Possible Improvements

Kmeans++ is a sophisticated algorithm that can find closeness between datasets of large dimensions. In this project, I resorted to applying kmeans on one dimension of data, namely the scores. Having more quantifiable attributes in the dataset would potentially enhance the accuracy of predictions. Another major improvement would be the use of bigrams in sentiment analysis. Having words with negations were not identified in my procedure. I did attempt to introduce it in the code to no avail.