Classifying Online User Behavior Using Contextual Data

Akshay Rampuria,[1] Anunay Kulshrestha,[1] Aditya Ramakrishnan[1]

[1]Stanford University
{rampuria, anunay, aditya94} @stanford.edu

December 13, 2014

In this paper, we investigate the problem of classifying user behavior using freely available user generated data like tweets on Twitter, reviews on Amazon and eBay etc. We follow the primary 25-label categorization that Amazon uses to bucket their products. By parsing a user's tweets, our algorithm attempts to predict which of the 25 categories the user is referring to. The learning model we present is based on a combination of a Multinomial Naive Bayes and Logistic Regression classifiers, which are augmented with an asymmetric distance metric and other classifiers based on contextual data sourced from Amazon and eBay. With these improvements, the latest iteration of our learning model attains an F1 score higher than 92%.

Introduction

Despite the great computational power of machines, there a some things like interest-based segregation that only humans can instinctively distinguish. For example, a human can easily tell whether a tweet is about a book or about a kitchen utensil. However, to write a rule-based computer program to solve this task, a programmer must lay down very precise criteria for this these classifications.

There has been a massive increase in the amount of structured user-generated content on the Internet in the form of tweets, reviews on Amazon and eBay etc. As opposed to stand-alone companies, which leverage their own hubs of data to run behavioral analytics, we strive to gain insights into online user behavior and interests based on free and public data. By learning more about a user's preferences and interests based on this parsed data from numerous heterogeneous sources, we can classify his/her interests. This problem of classifying online user behavior is especially interesting and perhaps complex because multiple labels can be assigned to the same tweet. For example, users may have tweeted about how much they liked reading the Lord of The Rings Trilogy, and then playing the game on their XBox. One could use this data to predict either sentiment, and given that tweets are at most 140 characters, the problem complicates.

Data

The relevant data was made available to us by our research advisors: Professor Ashutosh Saxena and Aditya Jami. They obtained the data using a web crawler which collected publicly available data and arranged it by user. This data was then split into three datasets :

1. Twitter Dataset (denoted by S): This data includes people's public tweets on particular products or services. The labels are in a nested-JSON format, where the depth of the JSON represents the degree or specificity of

classification. We have about 65,000 social conversations with content extracted and masked appropriately from Twitter to comply with their terms of service. We use a recursive algorithm to get the least specific labelling on a given datapoint im this data. We divide this Twitter data, cross-validating and using 70% to train and 30% to test.

2. Amazon Dataset (denoted by A): Each datapoint in this dataset consists of the review of an Amazon product and the metadata associated with it. Each instance is in the form of nested-JSON, where the depth of the JSON represents the degree or specificity of classification. Along with each review, we have data that helps us link the same users across different platforms, so that we can make predictions on their behavior. We have about 4 million reviews of approximately 400,000 users. We use data from this dataset to train our secondary classifiers and obtain metrics for comparisons.

3. eBay Dataset (denoted by E): Each datapoint in this dataset consists of the review of an eBay product and the metadata associated with it. Each instance is in the form of a similar nested-JSON, where the depth of the JSON represents the degree or specificity of classification. We have 35,000 eBay reviews and we use it, like we use our Amazon dataser- to train our secondary classifiers and obtain metrics for comparisons when the confidence of our Bayesian prediction is on the lower side.

Our initial data wrangling involved parsing the entire data-set and generating a map such that each key represented the broadest possible label for a datapoint. The value associated with each key is a list containing reviews or tweets about each of these parent labels. Making this structure involved recursively searching each datapoint for its least specific parent, and associating data points with the same label. For all this parsing, we heavily employed Python's `json` and `codecs` modules. Since, many of our datapoints had multiple broad labels, we randomly choosed one of the possible offerings and assumed that there was only one label for the rest of the problem. The code is structured in a way to be very easy customizable, that is, we could get any combination of the three datasets at our disposal by making simple function calls from the different modules.

Models

We sanitize every social conversation $s \in S$ by first removing all punctuation and then all stop words like at, the, is, which etc. These commonly used words do not add much information and can be ignored while building the feature vector. Additionally, we use stemming to reduce all words to their word root. For example, the words 'stemmer', 'stemming', 'stemmed' are all reduced to 'stem'. This greatly enhances efficiency by mapping all related words to the same root. Futhermore, we remove all links and usernames because they are not indicative of the class the tweet should belong to and thus, may confuse the classifier.

After this intial pruning, we use a bag-of-words model to obtain a term count vector for each social conversation in S. For this, we build a large vocabulary V of all words that occur in the dataset. The feature vector for $s \in S$ is $|V|$-dimensional and there exists a bijection between features and words in V. More precisely, the value of the i-th feature for the j-th document is the importance of the i-th word (indexed in V) to the j-th document.

The naive way to achieve this uses term counts of individual words in a document. But clearly, this does not account for commonly used words. To account for commonly used words and gauge how important a word is to a given $s \in S$, we replace the term counts by the tf-idf of each word in the feature vectors.

tf-idf (term frequency-inverse document frequency) is defined as

$$tfidf(w, s, S) = tf(w, s)idf(w, S) \tag{1}$$

where $tf(w, s)$ is simply the frequency of word w in s and

$$idf(w, S) = \log \frac{|S|}{|\{s \in S : w \in s\}|} \tag{2}$$

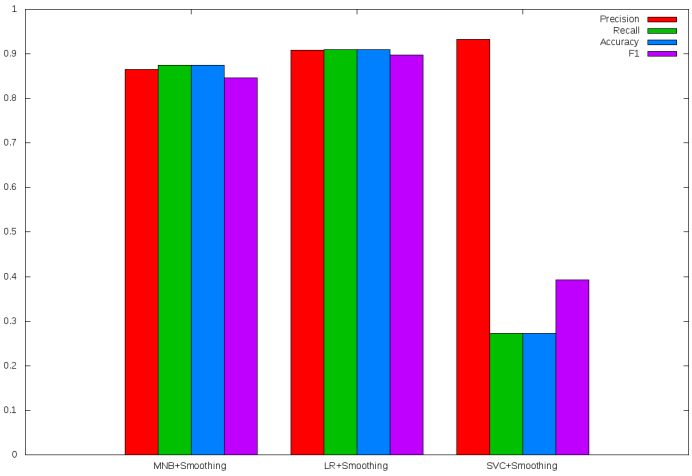
While we obtain satisfactory results ($\sim 71\%$ accuracy) with the naive definition of $tf(w, s)$, in order to improve the learning model we used smoothing of term frequencies such as

$$tf(w, s) = 0.5 + \log \frac{0.5 \times f(w, s)}{max\{f(w, s) : w \in s\}} \tag{3}$$

and

$$tf(w, s) = 1 + \log(f(w, s) + 1) \tag{4}$$

where $f(w, s)$ represents the frequency of word w in s. The feature vectors described above are then used to train multiple classifiers and results of cross-validation are depicted below. The smoothing significantly improved results (in case of Multinomial Naive Bayes and Logistic Regression).



Algorithms

The tf-idf vector thus obtained is the feature vector we consider for each $s \in S$. All this computation is done using the `nltk` and `sklearn` Python modules. After obtaining the feature vectors, we trained multiple classifiers on them. As evident from the graph above, Multinomial Naive Bayes and Logistic Regression produce the best results.

As a second and more novel approach, we augment the bag-of-words model described above by employing the eBay + Amazon dataset $K = (E \cup A)$. We prune each $e \in K$ by sanitizing it, removing stop words and computing the tf-idf vector as mentioned in the previous section. Note that now the vocabulary V contains text from both K and S. Thus, the tf-idf vector of each $s \in S$ is now different. Let $s \in S$ and p_i be the probability of s belonging to label y_i. In the first model, we simply predict

$$arg \max_i p_i \tag{5}$$

as the label for s. But in the second model, we compute $d(s, e)$ for a given s and every $e \in K$. Then denote

$$e_i = arg \max_{e, y(e)=i} d(s, e) \tag{6}$$

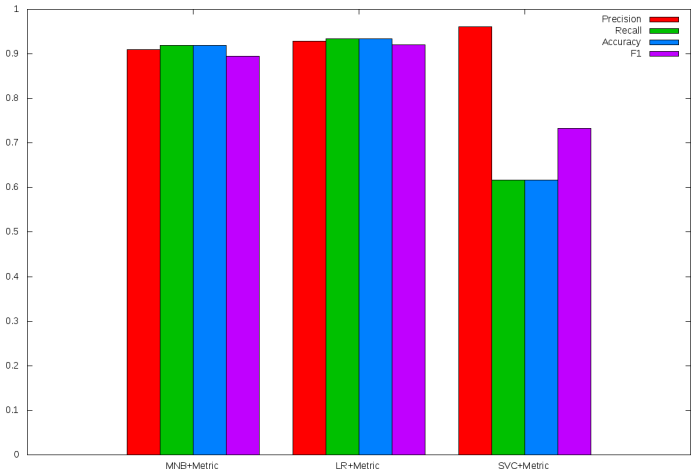
where e_i is the sample that minimizes $d(s, e)$ over all samples that have label i and d is a metric on $|V|$ dimensional tf-idf vectors s and e. The probabilities p_i are then augmented using e_i and d as

$$p_i' = p_i^{\alpha} d(s, e_i)^{1-\alpha} \tag{7}$$

where α is a heuristic parameter. We considered the asymmetric metric d defined as

$$d(s, e) = \frac{\langle s, e \rangle}{||s||^2} \tag{8}$$

For $\alpha = 0.6$, this metric further improves our learning model as depicted in the following graph.



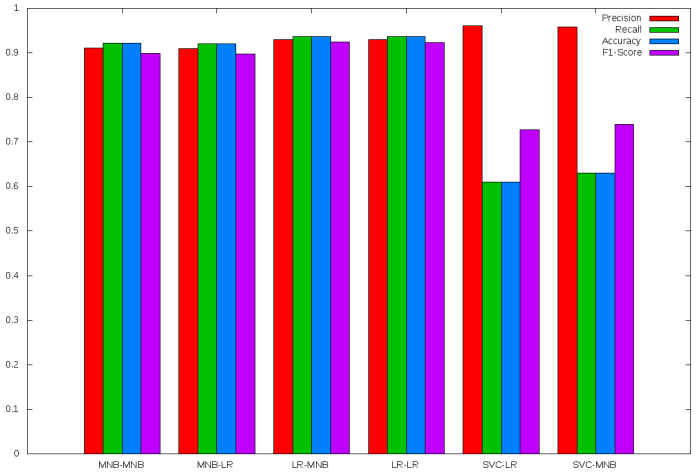
The third approach we explored included training another classifier on the contextual eBay and Amazon data sets and combining the posterior probabilities. We considered multiple pairs of classifiers out of which Logistic Regression on S and Multinomial Naive Bayes on $E \cup A$ provided the best results.

For every document s in the training set, we calculate the probabilities $p_y(s)$ and $p'_y(s)$ where p_y comes from the first classifier trained on dataset S while p'_y is derived from the second contextual classifier trained on datasets E and A. Here, y represents any given label.

To combine these, for a given s and y, we estimate a heuristic α such that the total probability of s belonging to y is modelled by

$$p_y(s)^{\alpha} p'_y(s)^{1-\alpha} \tag{9}$$

Using $\alpha = 0.3$ improves the model as depicted in the graph below.



Challenges and Future Work

Classifying contextual data has always been a challenge. In this project, we tried to use some context (from eBay and Amazon) to improve our learning model for classification of tweets. This represents a very small foray into a huge class of computationally and conceptually hard problems. Thus, there is immense scope for further work in this field.

One such direction can be learning the heuristic parameter α introduced in iteration 2 and 3 of the learning model. This will greatly improve the algorithm as hand-tuned parameters invariablly lead to overfitting of data as opposed to learning the same.

Another direction we would like to explore is the multi-level hierarchical classification problem discussed at the outset. While we only designed a learning model for the top-most layer of the label-tree, using similar techniques, we can descend into different labels, learn the parameters α and eventually design a holistic learning algorithm for every level of the tree.

Results

Iteration 1

Learning Model	Precision	Recall	Accuracy	F1-Score
MNB+Smoothing	0.864388723183	0.874775914795	0.874775914795	0.846692918541
LR+Smoothing	0.908452657545	0.909891384583	0.909891384583	0.897813207512
SVC+Smoothing	0.932119745059	0.273278498365	0.273278498365	0.392663198254

Iteration 2

Learning Model	Precision	Recall	Accuracy	F1-Score
MNB+Metric	0.909414754458	0.919223874301	0.919223874301	0.895260340266
LR+Metric	0.928628224442	0.934514394179	0.934514394179	0.920941350093
SVC+Metric	0.961527670924	0.616735210376	0.616735210376	0.732503810712

Iteration 3

Learning Model	Precision	Recall	Accuracy	F1-Score
MNB-MNB	0.910508288816	0.921491089318	0.921491089318	0.898334307175
MNB-LR	0.910080192626	0.92085837815	0.92085837815	0.897585711328
LR-MNB	**0.930290907704**	**0.936992512918**	**0.936992512918**	**0.923971244014**
LR-LR	0.929384383094	0.936043446167	0.936043446167	0.923096109637
SVC-LR	0.961482795215	0.609669935674	0.609669935674	0.727094477477
SVC-MNB	0.957732469337	0.630180322683	0.630180322683	0.739669170455

Clearly, in the third iteration of the learning model, the combination of a Logistic Regression classifier (on S) and Multinomial Naive Bayes classifier (on $E \cup A$) produces the best results on our data set (in bold above).

Acknowledgements

We would like to thank our research mentors, Professor Ashutosh Saxena and Aditya Jami. It was they who identified the problem of online user-generated text classification, and provided us the requisite data to tackle it. We met on a weekly basis to discuss our progress and they were extremely supportive in explaining intuitions and giving suggestions.

References

1. Abdulmutalib, Najeeb, and Norbert Fuhr. *Language models, smoothing, and idf weighting.* Information Retrieval (2010).

2. Rennie, Jason D., et al. *Tackling the poor assumptions of naive bayes text classifiers.* ICML. Vol. 3. 2003.

3. Kibriya, Ashraf M., et al. *Multinomial naive bayes for text categorization revisited.* AI 2004: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2005. 488-499.

4. Python libraries : `scikit-learn, numpy, scipy`