Sentiment Classification: Judging Attractiveness of People through Textual Descriptions from Online Reviewers

Natth Bejraburnin, Nathan Howard, John Le

December 12, 2008

Abstract

We aim to judge attractiveness of people given information provided by online reviewers. We were given the data from facestat.com and we developed and tested several machine-learning algorithms to meet these goals. The data is gathered from a variety of online reviewers, either anonymously or not. Among the information are single word textual descriptions, judgements on intelligence level, wealth, weight, gender, ethnicity. We mainly thought of our feature set as a bag of words. To improve upon the bag of words model we clustered the words based on several techniques, most notably mutual information, and then created a concept mapping from words that were less informative to words with high mutual information scores. This helped improve accuracy by about 0.50 percent with Naive Bayes and about 0.06 percent on Joachim's SVM light. We also tried logistic regression and locally weighted linear regression with varied success.

1 Introduction

Machine learning algorithms, such as Naive Bayes, have been used with great success for classifying text documents, particularly for classifying spam emails. Applying methods from the text classification problem, we wanted to predict the attractiveness of a person given their textual descriptions. We tested a couple of bag of words (BOW) model of the textual descriptions on several different algorithms, particularly regression, Naive Bayes and SVM's. The BOW model we tested was, the entire vocabulary of the training set, the 100 most common words, and the 100 words with highest mutual information score. Mutual information is a commonly used technique for feature selection which measures the amount of information that a value of one variable gives about another variable (Yang and Pedersen, 1997; Cover and Thomas, 1991 as cited in Schneider, 2004). And we calculated it with:

$$MI(x_i, y) = \sum_{x_i \in \{tokens\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)},$$

where x_i denotes whether token i appearing in a face, y denotes attractiveness. $p(x_i, y), p(x_i)$ and $p(y)$ can all be estimated from their empirical distributions on the training set. We will also describe a method where we used a given BOW model to cluster words and enhance our features.

2 Data Information

The textual descriptions came from www.facestat.com, particularly they are textual descriptions that are submitted by mostly anonymous reviewers. The non-anonymous reviewers' judgments were extremely sparse and hence we treated all reviewers as anonymous, though we had wanted to use algorithms, such as EM clustering, that analyzed biases within the reviewers. Attractiveness ratings were determined by averaging all of the reviewers judgments on a scale of 1 to 4, and we deemed that a person was attractive if their average attractiveness rating was greater than 2.5. This number was chosen objectively as it is the center of 1 and 4. The methods for gathering data justifies objectively choosing a standard for attractiveness that is independent of the actual judgments.

The textual descriptions are single word descriptions for a variety of categories, such as 'Describe in One Word', 'Age', 'Gender', and 'What is my worst feature?'. The person uploading the face gets to choose up to five categories they especially want judged, accounting for playful categories like 'How intoxicated am I?'. The only category which will always be judged by at least one reviewer is 'Describe in One Word'. The 'Age' and 'Gender' categories are inputted by the person uploading the face, although the user can also choose to have reviewers judge these as well. We worked on a subset, 11989 faces, of the data, where all of the faces have an attractiveness rating and some words in the 'Describe' category.

Judgments for each category are gathered from a pool of reviewers providing textual descriptions conforming to the category. The reviewers are either randomly selected from this pool of reviewers or randomly selected from people who anonymously view the site on the Internet. For each reviewer that is reviewing the categories they judge are independently randomly selected. Furthermore, while reviewing, reviewers will not be able to see the past judgments of a particular face unless by chance they had come across the face while browsing. Reviewers cannot choose to review a particular face, they can only decide that they want to review faces and from then on the website randomly gives them a face to judge. Because reviewers cannot see how other reviewers have described a particular face, we decided to objectively define attractiveness as having a rating greater than 2.5.

3 Pre-Processing

From the textual data described above, we generate numerical matrices for our learning algorithms by using the following rules:

1. Each image is represented by a row matrix whose dimension is 1 x 113.
2. The first entry is binary with 1 indicating that the person is attractive and 0 (or -1 if needed for SVM) otherwise.
3. The second entry is the average of all the numbers posted under category 'Age'.
4. The next 8 entries correspond to the categories of ethnicity. Each entry is the average of the counts on each category.
5. The 11th is the average of the counts of responses with 'male' (1) and responses with 'female' (0).
6. The 12th is the average judgment of weight, where weight, like attractiveness, is judged on a 1 to 4 scale.
7. The 13th is binary value describing whether the user wanted to be judged on wealth.
8. The last 100 entries are generated with the Multinomial Event Model i.e. the ith entry represents the count of the ith word's occurences in the set of judges' descriptions of this image.
9. Then only for Regression and SVMs, we normalize the matrix. For Naive Bayes, we use only the last 100 entries, for reasons which we'll explain later.

4 Methods

4.0 Clustering

We clustered words to the words we used in the BOW model. Our method was to cluster a non-cluster point to that point based on the metric $P(\text{cluster point appears on a face}|\text{non cluster point on a face}) =$

$\frac{\text{\# of times cluster point appears on a face a non cluster point appears in}}{\text{total \# of words on faces in which the non cluster point appeared}}$. Then we treated a non cluster point, i.e a token that was not used in our BOW model, as the token

$$argmax_{clusterpoint}P(\text{cluster point appears on a face}|\text{non cluster point i appears on a face}).$$

This means any time we saw the non cluster point that we would treat it as

$$argmax_{clusterpoint}P(\text{cluster point appears on a face} \mid \text{non cluster point i is on a face}).$$

We applied this feature adjustment method for the BOW model's of the 100 most common words and the 100 highest scoring mutual information words. This method did not need to be applied when we used all of the tokens in the BOW since in this case all words in the training sets were treated as cluster points. The motivation for this clustering approach is twofold. First, the internet jargon used by judges has both incorrect spelling and non-standard words. Furthermore the input doesn't allow reviewers to add spaces, hence there are a lot of tokens which are concatenations of several words, sometimes delimited by nothing. These factors make stemming/lemmatization less effective and thus we decided not to apply stemming or lemmatization to the words. An additional benefit of our mapping is that it maps semantically similar but lexically different words together. For example, in our BOW 'over-made' is mapped to 'fake', 'geezer' is mapped to 'old' and 'off-beat' is mapped to 'artsy'.

4.1 Naive Bayes

With Naive Bayes we would just use the words from the BOW model, we ignored the other categories. We at first took the other categories into account but they seemed to be introducing more noise and our naive bayes model was worse classifier. We tried looking at those objects as normally distributed averages and used them in our naive bayes model but this turned out to be unfruitful. We also tested treating the categories as tokens but from mutual information they did not turn out to be useful based on the mutual information metric, and hence we ended up ignoring them in for our final discussion. The results of the clustering applied to naive bayes appears in the charts below. Naive Bayes saw slight increases in test set accuracy and decreases in train set accuracy, and increases in precision and recall, when using the clustering model as opposed to not clustering. This appeared to tell us that clustering indeed helped improve our generalization error although as our chart shows not greatly and possibly by not a significant amount.

4.2 Logistic and Locally Weighted Linear Regressions

With the regression methods we plugged in the models to regression models. It took forever to run and the charts below describe our accuracy. In the chart For Locally Weighted Linear Regression, we ran it by testing several bandwidths and the chart shows us results from choosing the bandwidth that gave us the least test error.

4.3 Support Vector Machine

We use Joachim's SVM-light to implement the Support Vector Machine algorithms (SVM) on our datasets. We try this implementation with both the linear kernel and the Gaussian kernel, or radial basis function (rbf), whose Gamma parameter is determined by the Cross-Validation method described below. For Cross Validation, each candidate model is varied only by the Gamma parameter, all else fixed. Since most feature values range between -1 and 1, we need to use small Gamma in order to increase the gap between any two non-similar samples. Some trials suggest Gamma have the order of 10^{-3} and we define the candidate models as shown in the table. We train each of these on a set of 1000 samples and test it on a set of 500 samples. The results (see Table 1) recommend we use Gamma = 0.0045.

Then we run the selected model on two sets of data: the one with and without clustering. From the results (both generalization accuracy and precision/recall) displayed in the chart, we can see that clustering does not do significant improvement for SVM.

Table 1: Results from running Cross Validation

γ	0.0005	0.001	0.003	0.004	0.0045	0.005	0.008	0.01	0.5	1
Accuracy Rate	78.80%	79.00%	79.80%	79.80%	80.00%	79.80%	79.20%	79.40%	70.20%	70.20%

4.4 Comparison of the Algorithms' Performances

Algorithms	Training size	Accuracy Rate		Precision/Recall	
		w\clustering	w\o clustering	w\clustering	w\o clustering
	500	76.20%	76.20%	58.75%/63.95%	58.75%/63.95%
	1000	76.60%	76.00%	58.62%/63.27%	58.71%/61.90%
	1500	76.40%	75.80%	59.24%/63.27%	58.33%/61.9%
Naive Bayes	2000	76.40%	76.60%	59.24%/63.27%	59.49%/63.95%
	5000	77.40%	76.40%	61.04%/63.95%	59.34%/63.27%
	FULL	76.60%	75.80%	59.49%/63.95%	58.23%/62.59%
		Avg. 0.5% improvement			
	500	N\A	68.20%	N\A	N\A
	1000	N\A	72.40%	N\A	N\A
Locally Weighted Linear Regression	1500	N\A	73.40%	N\A	N\A
	2000	N\A	73.60%	N\A	N\A
	5000	N\A	73.60%	N\A	N\A
	FULL	N\A	75.00%	N\A	N\A
	500	N\A	73.60%	N\A	N\A
	1000	N\A	74.00%	N\A	N\A
Logistic Regression	1500	N\A	72.60%	N\A	N\A
	2000	N\A	73.20%	N\A	N\A
	5000	N\A	73.20%	N\A	N\A
	FULL	N\A	71.80%	N\A	N\A
	500	78.20%	78.60%	72.09%/42.18%	72.73%/43.54%
	1000	78.40%	78.60%	70.53%/45.58%	70.41%/46.94%
	1500	78.60%	78.80%	67.86%/51.70%	69.16%/50.34%
SVM with linear kernel	2000	76.90%	78.40%	69.91%/53.74%	66.39%/53.74%
	5000	79.40%	79.60%	69.64%/53.06%	69.91%/53.74%
	FULL	80.40%	79.80%	70.25%/57.82%	70.18%/54.42%
		Avg. 0.066% improvement			
	500	77.60%	79.00%	68.82%/43.54%	72.83%/45.58%
	1000	80.00%	78.80%	73.74%/49.66%	71.13%/46.94%
	1500	78.60%	78.60%	68.52%/50.34%	68.87%/49.66%
SVM with Gaussian kernel	2000	79.20%	78.60%	67.48%/56.46%	66.13%/55.78%
	5000	77.60%	78.20%	65.22%/51.02%	66.67%/51.70%
	FULL	78.20%	79.00%	64.62%/57.14%	66.94%/56.46%
		Avg. 0.166% drop			

4 Discussion

The most interesting outcome of our project was the bag of words clustering. While clustering did not decrease test set error significantly, its word associations proved interesting in their own right and worthy of further study. Since words exist in a discrete space and the spelling of two words usually gives little guidance to how they are related, there is no obvious metric of the distance between to two words. However, while simple, the results of our clustering introduces a topology on the words which, in many cases, correlate with an English speakers notion of the relatedness between words. With regards to the actual text classification, we found that a SVM performed better than a Naive Bayes classifier which performed better than logistic regression. These results are in accord with previous text classification trials (Kim, Sang-Bum et. al, 2006) which found that SVMs performed better than Naive Bayes classifiers. One hypothesis for the reason our Naive Bayes classifier performed relativly well against the SVM is that, in our case, the Naive Bayes's assumption that the probability of each word appearing is independent of which other words are in the text closely approximates reality since most the tags are only one word.

5 Future Work

1. Adjust the concept mapping: We currently pick a single informative word to map uninformative words. The word is chosen by picking the informative word that has the highest probability of showing up with our given uninformative word. This might give too much weight to an uninformative word being mapped to a word useful in determining attractiveness. Instead of doing this we would consider the following. If we see an uninformative word, then to the BOW feature vector for each good word we add the probabilities that the good word shows up with an uninformative word, calculated by (number of times good word appears in faces with the uninformative word)/(total number of words that appear in faces with the uninformative word). Intuitively this should smooth out the weight of adding a full instance of the informative word to the feature vector by distributing it across the clusters of words given by the BOW.

2. Facestat.com allows you to mark out the face you want judged. If the site could be expanded so that you can mark out features to be judged we maybe be able to use textual descriptions to find novel features within images. This could help generate models for what an attractive face might look like.

3. Define the topology of the space given by the words being points, and letting the distance metric be defined by probabilities that words show up together. We could use this to create a visualization for the space and if we color sections of the space based on attractiveness it could look pretty. Like if on the scale of colors ultraviolet means more attractive and as we move towards words that describe unattractive faces the color shifts towards the red end of the color spectrum.

4. From the comparison of the results from each algorithm we implemented, we can observe that the generalization accuracy does not exceed 80%. It could be expected that our training datasets are not linearly separable, as it contains around 20% outliers. One reason that accounts for this high number of outliers must be that the quality of attractiveness is not universal objective. As a consequence, there must be quite a number of images that possesses similar feature values to ones in the opposite category. The training datasets could be improved so that they reflect more truly the performance/effectiveness of the algorithms implemented against them.

6 Acknowledgements

We are thankful to www.facestat.com and particularly Lukas Biewald for for providing all the data.

References

[1] A. McCallum, "K Nigam - AAAI-98 Workshop on Learning for Text Categorization," 1998 - dns2.icar.cnr.it., http://ds.internic.net/rfc/rfc1738.txt; accessed August 23, 1997.

[2] A. Ng, "CS 229 Lecture Notes Autumn 2008."

[3] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng, "Some Effective Techniques for Naive Bayes Text Classification," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, Vol. 18, NO. 11, November, 2006.

[4] T. Joachims, "Making Large-Scale SVM Learning Practical. Advance in Kernel Methods - Support Vector Learning," B. Schalkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[5] S. Karl-Michael, "A New Feature Selection Score for Multinomial Naive Bayes Text Classification Based on KL-Divergence," *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computation Linguistics*, Association for Computational Linguistics: July, 2004, Page 186–189.