Fiona La, Daniel Pak, Long Luu
CIS 520 - Machine Learning
Final Project Report

## 1 Naive Bayes model

### 1.1 Performance on text data

The Naive Bayes model fits an empirical multinomial distribution to the text data. Let $Y$ = {0,1} indicate the tweet's mood and **X** = {1,2,...,10000} indicate the index of words in our vocabulary. Naive Bayes model finds the joint probability distribution $P(\mathbf{X}, Y)$ under the assumption that $X_i$ are independently conditioned on $Y$. In other words, Naive Bayes assumes that $P(\mathbf{X}|Y) = P(X_1|Y)P(X_2|Y)...P(X_{10000}|Y)$. Given this, the posterior probability distribution of each new sample **x** will be computed using Bayes rule $P(Y|\mathbf{x}) = P(\mathbf{x}|Y)P(Y)/P(\mathbf{x})$. The classifier will choose the MAP solution.

The performance of the model is pretty good. The cross-validation accuracy is around 80.3%. It demonstrates the robustness of Naive Bayes model when we have a small amount of data, like in this case (4500 training data and 10,000 features). That is the general characteristic of simple models like Naive Bayes. When the assumptions of simple models are not severely violated, the simple model may obtain good performance by a trade-off between the variance and the bias. Therefore, the good performance of Naive Bayes in this case suggests that its independence assumption is not too far from the true relationship between words. The result is also consistent with previous studies showing the superior performance of Naive Bayes. In Ng and Jordan (2002), the authors showed that Naive Bayes converges more quickly to the lower asymptotic error bound than the logistic regression when the number of sample increases. In other words, when the training data is limited, the generative models like Naive Bayes may perform better than the discriminative models like logistic regression.

### 1.2 Performance on image probability data

We can also use the Naive Bayes model for the image probability data in which each image in a twitter has a probability distribution over categories (1365 objects and scenes). When we used normal distribution to model the likelihood, the cross-validation accuracy is about 57% which is only slightly above chance. When we convert the image probability data to frequency format (i.e. *train img freq* = *round*(100□*train img prob*)) and fit a multinomial Naive Bayes, the performance is only slightly increased to 62%. The low performance of Naive Bayes in this case suggests that the independence assumption significantly deviates from the true distribution.

### 1.3 Data visualization

In order to have a visualization of the data under Naive Bayes model, we use the fit Naive Bayes model to calculate the posterior probability of hypothetical tweets containing only one word in the vocabulary. As a result we have the posterior probability of each word in the vocabulary $P(emotion|word)$. Then we take 25 words with highest 'sad' probability, 25 words with highest 'happy' probability and 25 words with the most neutral probability (closest to 0.5) and plot them in Figure 1. The red words are in class ’happy’ and the higher index as well as bigger size correspond to the higher ’happy’ probability. The blue words are in class ’sad’ and the higher index as well as bigger size correspond to the higher ’sad’ probability. The black words are the most neutral words and the closer it is towards the upper right corner as well as bigger size correspond to a probability closer to 0.5, that is, more neutral.

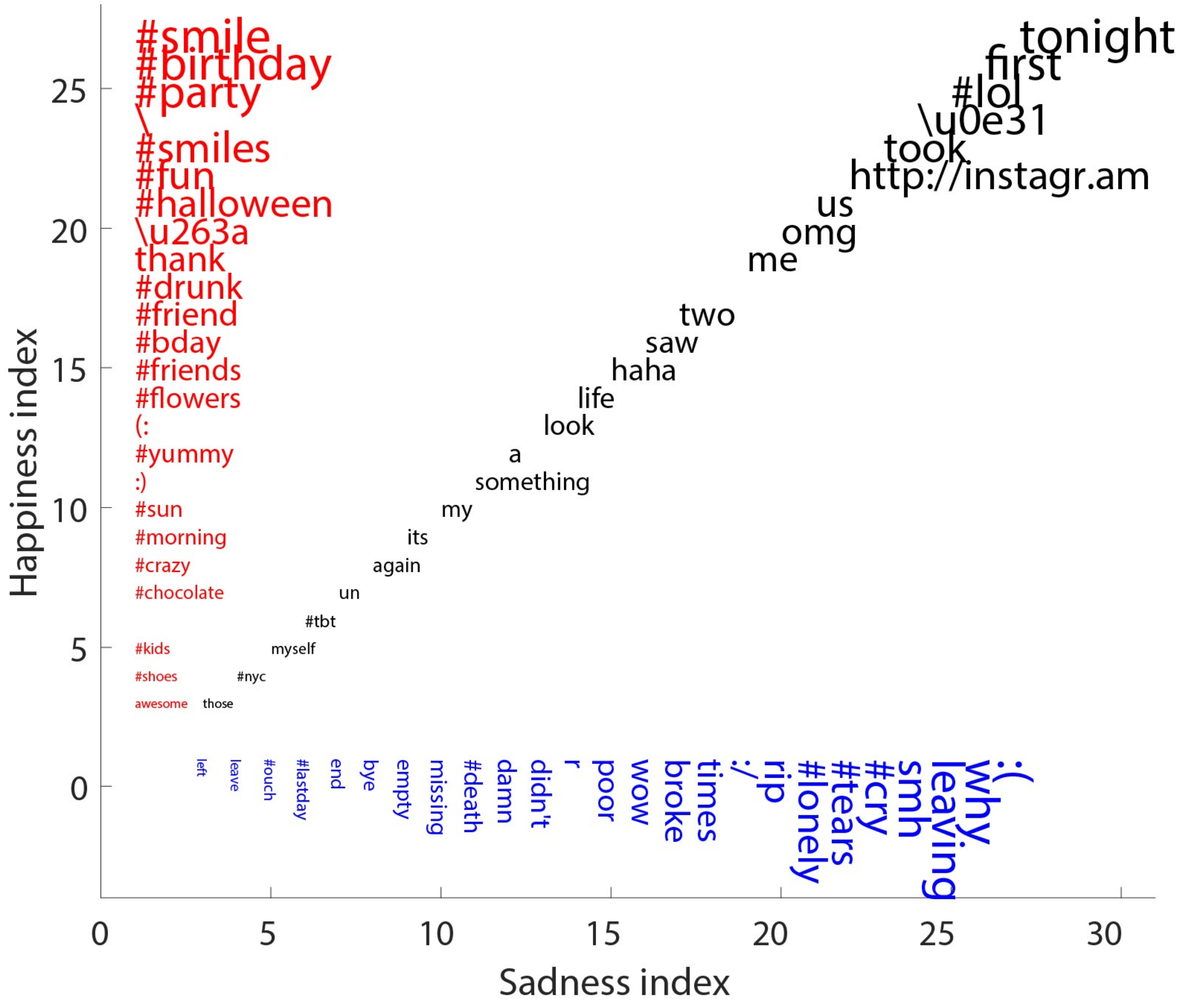

**Figure 1:** Visualization of data using Naive Bayes model

From the figure, it is more obvious why Naive Bayes performs pretty well despite its strong assumption on conditional dependency. Those words that have high posterior of ’happy’ class such as ’smile’, ’birthday’, ’party’, ’fun’ are strongly indicative of the emotion ’happy’. Similarly, the words with high posterior of ’sad’ class such as ’:(’, ’smh’, ’cry’ are very characteristic of the emotion ’sad’. Although it is less clear, the neutral words like ’tonight’, ’first’, ’took’ are more ambiguous in terms of the emotion. We also notice that some neutral words such as ’haha’ or ’lol’ should have been in the class ’happy’. One reason the Naive Bayes classifier doesn’t work well on those word may be because of the limited data we have on those words and accordingly the model will put more weight on the prior which is around the neutral probability 0.5.

## 2 K-Nearest Neighbors Model

K-nearest neighbors was the instance based method used to classify based on the word count data. First, different values of K were explored, as shown in table 1. Then, different norms were considered, table 2. The highest performing model had a K value of 1, with the trend of the higher the K value, the lower the 10-fold cross validation accuracy. Because K-NN is a non-parametric method, it is more flexible to the noise contained in the test set, making it more susceptible to overfitting the training set. In an effort to combat this issue, principal component analysis was performed to reduce the number of dimensions of the feature space. The optimal number of principal components to use was determined by how many were required to achieve 90% reconstruction accuracy; this value was 638.

**Table 1**. 10-fold cross validation accuracy for varying values of K with 638 principal components and the L2 norm

| **K** | **10-fold Cross Validation Accuracy** |
|---|---|
| 1 | 64.60% |
| 3 | 55.77% |
| 5 | 55.26% |
| 10 | 55.17% |
| 50 | 55.15% |

Another method to combat overfitting the training data would be to use a K value that is large enough to minimize the noise, but small enough to not include too many samples from the other class. But the 10-fold cross validation accuracy was highest for K = 1 and subsequently lower for K = 3 and K values greater than 3; suggests that similarly classified examples are not necessarily clustered together. This observation is confirmed by the poor results of the K-means algorithms we experimented with. From 2 clusters to 25 clusters, accuracies were poor, with 10-fold cross validation accuracies ranging from 0.50 to 0.56.

**Table 2**. 10-fold cross validation accuracy for different norms, K = 1 and 638 principal components

| **Distance Metric** | **10-fold Cross Validation Accuracy** |
|---|---|
| L1 | 65.11% |
| Linf | 62.60% |

With respect to our goal and to other models, changing the norm that 1-NN took did not dramatically affect the results. L1 performed slightly better than L2 while Linf performed worse. That they are not significantly different is not surprising, as the evidence found earlier suggests the distances between an example and its closest neighbors is not highly correlated with whether they are in the same class or not. Overall, K-NN was one of the worst models for classifying tweets as joyful or sad; in addition, the model was very large and classifying time was long due to the fact that each new example would need to be compared to the training points.

## 3 SVM model

We fit a SVM classifier in liblinear library to the text training data. The SVM model will find the hyperplane that maximizes the margin with the support vectors. In this case, because we cannot have a perfect linear separation, we have to use a regularized version of SVM in which we use a hinge loss to penalize the data points that are on the wrong side of the margin. We tried three regularized versions: L2-Regularization, L2-Loss; L2-Regularization, L1-Loss; L1-Regularization, L2-Loss. Moreover, for each regularized version, we vary the weight of the regularization term to observe the model's behavior. Figure 2 shows the cross-validation accuracy of the SVM classifiers for different regularizations.

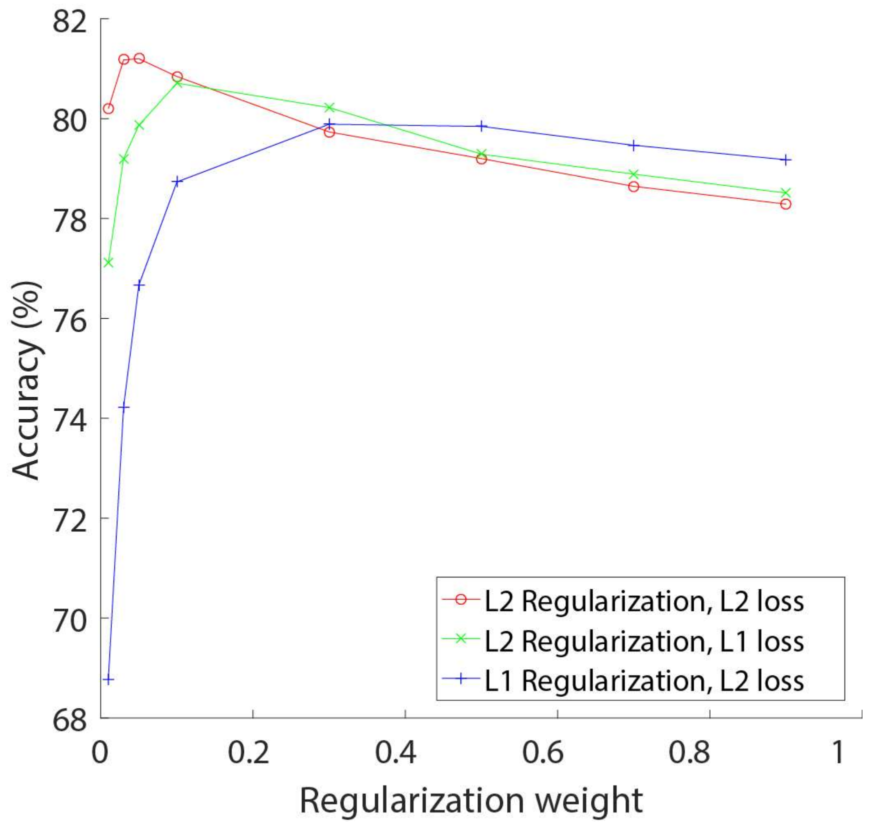

**Figure 2:** Cross-validation accuracy of the SVM classifiers for different regularizations

The first noticeable point is that there exists a sweet point on the accuracy curves corresponding to a regularization weight that results in good performance. A regularization weight that is too small (0.01) or too large (0.9) will lead to inferior performance. Second, it seems that L-2 regularization and L-2 loss works better than L-1 regularization and L-1 loss. It makes sense because in this case, we have many features (10,000) compared to the training data (4500). Therefore, a stronger regularization (L-2 instead of L-1) is more likely to work better. That is consistent with the fact that the best regularization weight of L2-regularization, L2-Loss (0.07) is smaller than the best regularization weights of L2-regularization, L1-Loss (0.1) and L1-regularization, L2-Loss (0.3).

The best accuracy of SVM is comparable and slightly higher than the accuracy of Naive Bayes which shows that with proper regularization, the discriminative model may outperform the generative model even when the data is limited. However, we have to fine-tune the parameters of the SVM model whereas there is no free parameter to fine-tune in Naive Bayes. Also, the Naive Bayes tends to run faster than the SVM.

## 4. Dimensionality reduction methods

We tried three main types of dimensionality reduction methods: word stemmer, Principal Component Analysis (PCA), and Latent Dirichlet Allocation (LDA). Since our main dataset (word counts) contained more features than observations, the goal was to reduce the number of features while losing minimal information so that supervised learning methods such as SVM and logistic regression would perform better. We used built-in MATLAB functions for training the models in this section.

Word stemmer is a process where similar words such as “absolutely” and “absolute” can be combined together to a unifying stem “absolut” such that the number of features can be reduced. In our case, the number of features after stemming, taking out hashtags, and manually combining some non-word features (such as :), :D, ^.^, ^_^, etc.) reduced from 10000 to 6772 (Lopez, 2006). By simply applying word stemmer, we were not able to improve performance very much. The cross-validation accuracy with logistic regression returned 0.8116 for no stemmer (full word count labeled data), and 0.8118 with stemmer. We believe that the word stemming alone did not increase performance because the dataset is already very sparse, which makes the effect of reducing the number of features negligible.

PCA looks for orthonormal bases that most accurately represent the data matrix. With the new word counts output from word stemmer, we used PCA to extract the eigenvectors with 400 highest eigenvalues for further testing. The eigenvectors were calculated with combined training data, which is vertically concatenated labeled and unlabeled training data. The scores for the labeled data were then calculated by subtracting the column mean of the combined data and matrix-multiplying the eigenvectors. The number of components was chosen based on performance of cross-validation with SVM. The cross-validation accuracy of SVM without PCA (full word count labeled data) was 0.7622, and 0.7913 with PCA (400 components). With fewer than 400 components, the accuracy dropped probably due to excessive loss of information from dataset, and with more components, the accuracy dropped probably due to overfitting or lack of complexity in the model (not complex enough for number of features).

LDA assigns topics for each document by looking for words that tend to appear in similar contexts throughout the entire input matrix. The number of topics and the number of iterations determine the complexity of the resulting feature map, and there are two hyperparameters that determine other variables such as learning rate that affect the performance of the output matrix (Steyvers, 2011). By hand-tweaking of values and cross-validation with logistic regression, the best number of topics was determined to be 100 and the number of iterations 200. The hyperparameters were tweaked slightly from the suggested values from the toolbox. The cross-validation accuracy of logistic regression without LDA was 0.8071 and with LDA was 0.7358. We believe that this decrease in performance was due to the sparsity of the dataset, limited number of observations, and loss of information. Specifically, since a topic is assigned to each word, the amount that a single word contributes to determining the sentiment of a tweet is in some ways normalized; a smiley face will no longer be inputted as a smiley face into the classifier, but instead a topic that includes other features that may not have as strong of an indication, which would hurt the performance of the classifier.

## 5. Ensemble method

Fiona La, Daniel Pak, Long Luu
CIS 520 - Machine Learning
Final Project Report

Our final model is a late fusion ensemble method that combines logistic regression and Naive Bayes models with only the word-stemmed word count matrix as input data. Both the pre-processing method and the two classifiers were chosen based on performances with cross-validation. Any processing other than word-stemming decreased performance, and other classifiers had accuracies around 0.01 less than logistic regression and Naive Bayes. We used the built-in MATLAB functions for the two models.

For logistic regression and Naive Bayes, we were able to get the posterior probabilities of the observations being classified as 0 or 1, instead of the actual classification estimates. Since these are binary classifiers, we simply took one column for each classifier (since the two columns add to a probability of 1) and trained a linear regression with the real classification values so that we have two parts of the model: one for converting a set of observations into posterior probabilities, and another for converting the posterior probabilities into classification estimates. In order to prevent overfitting the data, we divided the labeled training set into 10 folds, used 9 for training the logistic regression and Naive Bayes models, and used the last fold for training the linear regression with posterior probabilities and the true classification values. This ensemble method increased the performance by 0.007 from a simple logistic regression. We believe this model was successful because the two models performed pretty well to start with, and any ambiguity in one model had the potential to be corrected by the other model.

**Table 3.** 10-fold cross validation accuracies of all five methods

| **Method** | **10-fold Cross Validation Accuracy** |
|---|---|
| Naive Bayes | 0.803 |
| K-NN | 0.6511 |
| SVM | 0.809 |
| Dimension Reduction Methods | 0.8118 (word stemmer), 0.7913 (PCA), 0.7358 (LDA) |
| Ensemble Method | 0.8183 (cross-validation), 0.8098 (leaderboard) |

## Reference

Ng, A. & Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, *14*, 841.

Steyvers, M. (2011). Matlab Topic Modeling Toolbox 1.4. http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

Lopez, J. C. (2006). The Porter Stemming Algorithm (MATLAB version). https://tartarus.org/martin/PorterStemmer/