# Text based categorization predictions from book reviews

Fucheng Gao
Department of Computer Science and Engineering
University of California, San Diego
A53095931
fugao@eng.ucsd.edu

Pengfei Chen
Department of Computer Science and Engineering
University of California, San Diego
A53106511
pec028@eng.ucsd.edu

## ABSTRACT

Automatically categorization is of great significance nowadays where massive data flows to us every day, especially for companies receive and need big data. It is especially important when there are some data unclassified or unlabeled, how can we define the categorization is very challenging. What we do is to categorize a review text into some pre-defined categories based on its text features. We did some preprocessing and feature processing to improve the accuracy. And we tried several methods, either basic or advanced, including Linear Regression, KNN, Decision Tree, Neural Network and Nonlinear SVM. Then we compare these methods and analyze the results. Finally we come up with a conclusion.

## Keywords

Data mining, book reviews, neural networks, SVM, decision trees.

## 1. INTRODUCTION

The goal of our project is to predict the categories for a book review from the data set of book review data. We came up with this idea when we recall that when we are doing assignment 1, when we wanted to extract the categories as a feature, we noticed that not all book reviews have category classifications. But it's common sense that every book should have at least one category. So we want to predict the categories based on the these data.

We first did some preprocessing to extract features in Section 3. And then did feature processing to improve the accuracy in Section 5. We tried several methods, either basic or advanced, including Linear Regression, KNN, Decision Tree, Neural Network and Nonlinear SVM as in Section 6. Then we compare these methods and analyze the results. Finally we come up with a conclusion in Section 7.

## 2. DATA SET

### 2.1 Data set chosen

We reuse the book review dataset in assignment 1, which is a data set about book reviews. In the process of studying of assignment 1, when we wanted to extract the categories as a feature, we noticed that not all book reviews have category classifications. But in common sense, each book should have its own category. So we decided to build a data processing system to help users define the categories of books according to the review texts they've written.

Each datum consists of 9 fields, including:

'itemID', 'rating', 'reviewText', 'helpful', 'unixReviewTime', 'category', 'reviewerID', 'reviewTime' and 'summary'.

What we want to do is predict the categories of books based on the review texts.

In our assignment 2 project, the dataset are lines of data with category (in train.json.gz, approximately **120k** samples).

### 2.2 Exploratory Analysis

First we count the all the categories and see the distribution. We can see from the figure 1 that there are many categories, while the top categories appear very often and the other appear so rarely.

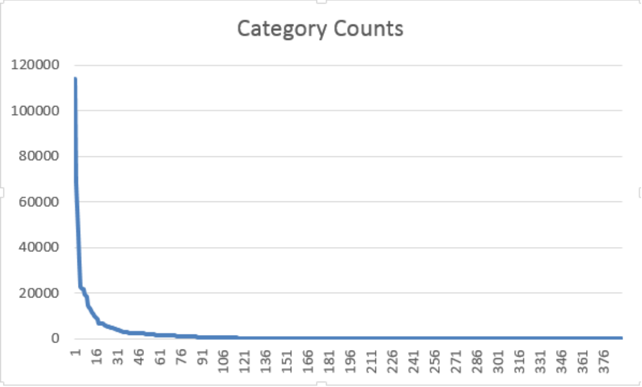

**Figure 1. Distribution of categories sorted.**

Then we can see from figure 2 about the top 20 categories. The last category, which is 'Anthologies' still has a count of 6,000 out of the total 120,000 reviews, counting for 5% of the total number. So this is sufficient for further processing.

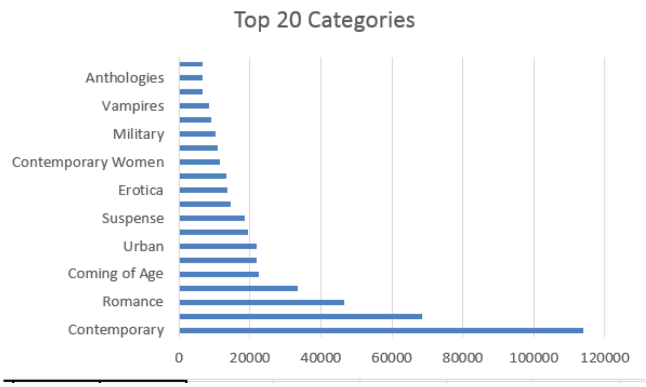

**Figure 2. Distribution of top 20 categories sorted.**

## 3. PREDICTIVE TASK

Our main work is to predict the categories based on the review text.

### 3.1 Data pre-processing

Apart from those reviews with no categorization, there are also some issues about categorization in the reviews.

The category field consists of several lists of strings, such as “books”, “children’s books”, and so on. At first we treat each string as a category, then merge all of the lists into one category list. In this process, we delete duplicate string. But after this process, we found many nonsense categories, such as “books” (which seems to appear in every sample), “Kindle Store” and “Kindle eBook” (which usually appear together, classify them separately seems nonsense).

To find out what’s going on, we visited the amazon.com which seems to be the data source. And it turns out that the ‘list of categories’ seems to be the hierarchy categories. So we discard all categories except the last category in each list. Because this is the category on the lowest layer. In other words, this categorization is the most specific one and thus best describes the book. For example, if there’s a category list like “Books => Education => School & Teaching”. Then we will process it into “School & Teaching”.

What’s more, there are still more than 300 categories, with the last 280 categories less than 5% of the total book reviews. In order to ensure the accuracy of predictions, we discard these categories.

**Table 1. The top 20 categories**

| | | | |
|---|---|---|---|
| Contemporary | Literature & Fiction | Romance | New Adult & College |
| Coming of Age | Romantic Comedy | Urban | Paranormal |
| Suspense | Women's Fiction | Erotica | Paranormal & Urban |
| Contemporary Women | Romantic Suspense | Military | Fantasy |
| Vampires | Series | Anthologies | Werewolves & Shifters |

So for various reasons, we first do some pre-processing to the raw data. The pre-processing includes three parts:

**i. Process the category field of each datum as we stated above to discard all the unnecessary fields.**

**ii. Compute count for every categories. Discard categories except for the top 20 categories with most count. (Table 1)**

**iii. Discard all samples without a categorization.**

## 3.2 Evaluation model

Given this classification prediction is a multiple classification, which means that a book can be classified in two different categories. Thus the evaluation criteria is not something like **MSE** (Minimum Square Error) or **MAE** (Minimum Absolute Error). We need to both look into the false negative errors and false positive errors.

In order to evaluate the accuracies of different models we use, we use four main criteria to evaluate the models. These four criteria are: **FPR** (False Positive Rate), **FNR** (False Negative Rate), **BER** (Balanced Error Rate) and **ACC** (Accuracy).

$$\mathbf{FPR} = \mathbf{False\ positives\ /\ \#labeled\ positive}$$

$$\mathbf{FNR} = \mathbf{False\ negatives\ /\ \#labeled\ negatives}$$

$$\mathbf{BER} = \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{FPR} + \boldsymbol{FNR})$$

$$\mathbf{ACC} = \frac{\mathbf{True\ positives} + \mathbf{True\ negatives}}{\mathbf{\#labeled\ negatives}}$$

After the pre-processing, there are about **120,321** samples. We divide the data into three parts:

a) Training data set. This includes the datum 1 ~ 70000.

b) Validation data set. This includes the datum 70001 ~ 100000.

**c)** Test data set. This includes the datum 100001 ~ 110000.

## 3.3 A simple baseline

In order find a standard for later complex models, we start by define a simple baseline of this prediction.

We start by the idea that some specific ‘functional’ words can help us categorize. Like what we do in homework 1, if a review text consists of the word ‘Wizard’ or ‘Witch’, we classify it into ‘Fantasy’ books. We want to define such two words for each category.

It is realistic to define these words by coming out of nothing, so we let program to do this thing for us. For each category separately we find all the average words count within the reviews of this category and average words count within the reviews not of this category. Record these two count as $\boldsymbol{C_t}$ and $\boldsymbol{C_f}$. We find two words with $\boldsymbol{max}(\boldsymbol{C_t} - \boldsymbol{C_f})$, and define these two words as the ‘representing’ words for this category. This is the idea that if a word is more likely to appear in such category, and less likely to be in the text not in such category. Then this word maybe somehow have some connection with this category.

So when we are about to predict a review’s categories. We check for each category if the ‘representing’ words of this category is in the review text.

The performance of the baseline will be discussed in details in later sections.

## 3.4 Assessing validity of the predictions

One standard for assessing the validity of our predictions is the baseline we stated above. We will compare the FPR, FNR and BER of our prediction to the baseline. Another way is to analyze the ratio of FPR and FNR, whether a too high FPR or too high FNR is not acceptable.

## 4. RELATED WORK

There are many related work on categorization base on text. In [1] talks about the automated categorization of texts into some predefined categories. This research to this problem is based on some advanced machine learning techniques. First a general inductive process automatically builds a classifier by learning from a set of preclassified documents, and the characteristics of the categories. This survey discusses the main approaches to text categorization using the machine learning methods. They discussed mainly about the following three problems: document representation, classifier construction, and classifier evaluation.

In [2], the book first discusses about the Classification and Regression Trees algorithm. This book also presets some examples from Richard Olshen's experience working in the Medical School at UC San Diego, and the method is widely used over the past 20 years.

In [3], there is one more book talks about classification and regression trees. But this book stands on the position that fifty years have passed since the publication of the first regression tree algorithm. So this book more talks about new techniques that have added capabilities that far surpass those of the early methods. The idea is that modern classification trees can partition the data with linear splits on subsets of variables and fit nearest neighbor, kernel density, and other models in the partitions. So regression trees can fit almost every kind of traditional statistical model, including least-squares, quantile, logistic, Poisson, and proportional hazards models, as well as models for longitudinal and multiresponse data. This article surveys the developments and briefly reviews the key ideas behind some of the major algorithms.

What's more, in [4] and [5], there are similar data sets that we are using for categorizations. In [7] and [8], introduces some of the most advanced methods for categorizations. In [9] discusses a way to reduce the dimensions in features which we are going to use in later sections.

## 5. FEATURES

Text cannot be directly categorized by a classifier. So we need to find a way to extract feature from each review text, and use the same format feature for classification.

As we stated before, we use the words in the review text to predict the categories. Given that there are more than 20,000 different words, it is not reasonable to taken all these 20k words into account. So we need to cut down the features before we run our models.

### 5.1 Semantic processing

First of all, many words like the pronouns, particles and prepositions definitely won't help us make predictions since they are 'meaninglessly' appearing in every sentence. So we use a stop-words list to list out all the words abandoned. Then we also do the stemming. For words like 'logic', 'logistics' that have the same roots we treat them as the same because they mainly have the same effect in the sentences.

This reduces the different words number to about 18,000.

### 5.2 Unigram selection

Commonly our approach is the bag-of-words model, which defines a fixed length vector of appearance of words in a specific dictionary. To build this model, we count the times of appearance for each word.

Since classification of different categories may be affected by different words, so we decided to select unigram for each category separately. At that time, for a specific category, we give samples label {0, 1} which stands for belonging to this category or not. And for unigrams vector, we have a **TF** (Term Frequency) feature of the samples.

By applying semantic processing, we have reduced some of meaningless words. Then, we need to further decrease the words number.

By observation, we noticed that many words with very few appearance are not likely to be meaningful words, mostly are some random meaningless combination of characters and numbers (this may be due to the word segmenting progress or misspelling). And these words may even cause overfitting issue in our classifier. So we retain the 2000 most frequent ones and throw away the others, which appear at most 150 times in 120000 review texts.

### 5.3 Chi-square method

As we said before, we want to select different words for different categories. For example, appearance of 'Children' is a powerful feature for "Children's book", but less effective for "Literature books". To find this relationship between feature and category, we used chi-square method.

**Table 2. The top 20 categories**

| | **Belong to category** | **Not belong to category** |
|---|---|---|
| **Word appears** | **A** | **B** |
| **Word doesn't appear** | **C** | **D** |

(For a specific category) For each feature (word), we divide sample set into 4 parts. (See Table 2 above.)

We define chi-square value as follows:

$$\chi^2(\boldsymbol{word}, \boldsymbol{category}) = \frac{\boldsymbol{N}(\boldsymbol{AD}-\boldsymbol{BC})^{\boldsymbol{2}}}{(\boldsymbol{A}+\boldsymbol{C})(\boldsymbol{A}+\boldsymbol{B})(\boldsymbol{B}+\boldsymbol{D})(\boldsymbol{C}+\boldsymbol{D})}$$

Which can be simplified to:

$$\chi^2(\boldsymbol{word}, \boldsymbol{category}) = \frac{(\boldsymbol{AD}-\boldsymbol{BC})^{\boldsymbol{2}}}{(\boldsymbol{A}+\boldsymbol{B})(\boldsymbol{C}+\boldsymbol{D})}$$

Since that:

$$\boldsymbol{N} = \boldsymbol{A}+\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{D}$$

$\boldsymbol{A}+\boldsymbol{C}, \boldsymbol{B}+\boldsymbol{D}$ are same for all words.

By using this method, words owning larger values means they have stronger relationships with this category (whether negative or positive). Then we sorted the words by chi-square and choose the first 100 words as final feature (separately for each category).

### 5.4 Bigram selection

Although stop-words such as "the", "not" usually don't convey any information. But when they are combined with the next word, they can convey meaningful information. So we adapt bigram model to catch these most meaningful bigrams. In this progress, we first build a dictionary for all bigram start by stop-words. And then choose 500 most frequent ones. After that, we adapt chi-square on these bigrams for all categories, and choose 20 most influential for each category.

### 5.5 PCA and LDA

Both LDA (Linear Discriminant Analysis) and PCA (Principal Component Analysis) are the most commonly used dimensionality reduction techniques in pattern-classification and machine learning applications. The goal of these methods is to project a dataset into a lower-dimensional space with good class-separability in order to avoid overfitting ("curse of dimensionality") and also reduce computational costs.

In practice, PCA finds the axis with maximum variance for the whole data set where LDA tries to find the axis for the best class separability. So in our project, we choose LDA method to compress the dimension of feature. And to do this step, we use the LDA function in both scikit-learn and Matlab Toolbox for Dimensionality Reduction for calculation in python and Matlab.

# 6. PREDICTION MODEL

## 6.1 Linear regression

We start by applying the common method, linear regression. Since linear regression is so familiar we are not going to introduce the definition and the common things, but state something we do.

### *6.1.1 Feature Adjust*

In order to have better access to these specific data, which is of a great number of features. While we still use the bag-of-words model, we tried to get better features as we stated in Section 4. So after our processing, there are about 2,000 features.

### *6.1.2 Result Adjust*

Since this is a multiple classification. It is not reasonable to assign the value from 1 to 20 to the categories. So we use a list of 20, each stands for a particular category. If a review is of a category, the field is labeled '1', otherwise '0'. And the predictions are floats. So we judge by comparing the prediction of such field with 0.5.

## 6.2 K-Nearest-Neighbors

K-Nearest-Neighbors (known as KNN) is a useful method for classifications. The main idea is to find the most $\boldsymbol{k}$ similar nodes in the training data with the one we are to classify. And classify by counting which classification these k-neighbors mostly belong to. In this situation, where each book review can have multiple categories. So for each category, we decide by if more than half of the k-neighbors are in such category.

### *6.2.1 Similarity Definition*

Since there are too many features for each review – even after dimension reduction – there are more than 2,000 features. And different words should have different weights of effects on different categories. What's more, we also should not consider the length of the review which doesn't apparently varies from category to category.

So it is not reasonable to use the similarity measurements like Euclid Distance or Manhattan Distance and so on. Instead, we use the cosine similarity based on the tf-idf to measure the similarities between all pairs of reviews.

## 6.3 Decision Tree

In some problems, features are nominal data, which has discrete values. For example, the taste of beer can be sour, like banana, and so on. We only use one dimension of value to express taste. And if there are many possible value, use (0, 1) feature to represent for each one will result in the scale of training data unreasonably increasing.

In this situation, decision tree is a method worth trying. In our model, we use the most frequent unigrams combined with some bigram starting with stop-words. (As we talk about in Section 5) Then, each feature appear/not appear can be treat nonmetric. Although in this situation decision tree is not as suitable as former beer test example, but also worth trying.

### *6.3.1 CART Algorithm*

#### *6.3.1.1 Impurity*

If we have one feature, when the feature is 0, the output is always 0, and if the feature is 1, the output is always 1. Then after we divide sample set by this feature, the samples in each subset should have same label. In this idealized example, the impurity of each subset is 0. In the root of decision tree, there are all the samples. And for each node of decision tree, we divide the samples in this node into 2 or more subsets according to one feature and gain lowest impurity of each side.

In our model, we define impurity as:

$$\boldsymbol{i(N) = P(c)P(!c)}$$

This means for any one category (each time we train classifier for one category), the impurity is product of positive rate and negative rate in the set.

#### *6.3.1.2 Greedy Algorithm*

In each loop, we look at all leaf nodes. For sample sets assign to each leaf, we try to find feature to divide them into 2 parts, and maximize

$$\Delta i(N) = i(N) - P_L\, i(N_L) - (1 - P_L)\, i(N_R)$$

The N is original set, the NL NR are subsets, and PL is proportion of node in NR, 1 - PL is proportion of nodes in NR. And we find divide with max delta $i(N)$ among all nodes, adapt it and generate new tree as initial of next loop.

#### *6.3.1.3 Stop criterion*

The criterion is generated by validation and cross validation. In all the 120000 samples, I select 80000 as train data, 20000 as validation data and 20000 as test data. In each loop I train the tree use train data. And then test the error rate on validation set. Once one node error rate doesn't decrease, then the node doesn't branch then. When all the leaf nodes stop branch, stop train procedure.

#### *6.3.1.4 Pruning*

Usually when tree is grown fully, we will meet overfitting. To avoid this issue, we adapt cost complexity pruning in our method. After the tree is fully grown, generate a series of subtrees$\{\boldsymbol{T_0}, \boldsymbol{T_1}, \boldsymbol{T_2} \ldots. \boldsymbol{T_n}\}$, while $\boldsymbol{T_{i+1}}$ is always generate from $\boldsymbol{T_i}$, and $\boldsymbol{T_n}$ is the root alone. At step *i* the tree is created by removing a subtree from tree *i-1* and replacing it with a leaf node with value chosen as in the tree building algorithm. Define the error rate of tree ***T*** over data set ***S*** as ***err(T,S)***. And the subtree that minimize

$$\frac{\boldsymbol{err(prune(T,t),S) - err(T,S)}}{\boldsymbol{|leaves(T)| - |leaves(prune(T,t))|}}$$

will be chosen. ***Prune(T,t)*** defines the tree gotten by pruning the subtrees t from the tree T.

## 6.4 Neural Network

Inspired by biological neural networks, artificial neural network is designed to model nonlinear mapping relationship between input and output. ANN usually perform as a system of interconnect nodes (which called "neural"). And in backpropagation neural network, there usually 3 layers of nodes: input layer, hidden layer and output layer. Nodes in adjacent layers have connection, while nodes in the same layer doesn't connect. Nodes in hidden layer and output layer receive input from former layer with assigned weight, and then turn to output value by sigmoid function.

### *6.4.1 BP Neural Network*

#### *6.4.1.1 Nodes*

In our NN model, we select 1000 features from unigram and bigram in the text. So the dimensions of input layer is 1000. And commonly we set hidden layer with 2000 nodes.

#### *6.4.1.2 Initial status*

We initial the weight matrix between each two layers with random value [-0.1, 0.1].

#### *6.4.1.3 Normalization*

Before training, we normalized the data of tf-idf score

#### 6.4.1.4 Training sample

We used 80000 samples from the dataset to train our neural network. The input for each sample is tf-idf score after normalization. And the output is whether this book belong to each category (30 dimensions of category in total)

#### 6.4.1.5 BP loop

Select sample $\boldsymbol{x} = [\boldsymbol{x_1}, \dots, \boldsymbol{x_n}]^{\boldsymbol{T}} \in \boldsymbol{R^n}$ and expected output $\boldsymbol{D} = [\boldsymbol{d_1}, \dots, \boldsymbol{d_m}]^{\boldsymbol{T}} \in \boldsymbol{R^m}$

Calculate output $\boldsymbol{Y} = [\boldsymbol{y_1}, \dots, \boldsymbol{y_m}]^{\boldsymbol{T}}$under the input

From output layer, adjust the weight matrix

$$\boldsymbol{w_{ij}^l(t+1) = w_{ij}^l(t) + \eta\delta_j^l w_i^{l-1}, j = 1, \dots, n_l, i = 1, \dots, n_{l-1}}$$

For the output layer:

$$\boldsymbol{\delta_j^l = y_j(1-y_j)(d_j - y_j), j = 1, \dots, m}$$

For the intermedia layer:

$$\boldsymbol{\delta_j^l = x_j^l(1-x_j^l)\sum_{k=1}^{n_{l+1}} \delta_k^{l+1} w_{jk}^{l+1}(t), j = 1, \dots, n_l}$$

Modify weight for all the samples

Re-predict the output of all the samples, and calculate the error rate. If error rate below the stop criteria (20%), stop BP loop

#### 6.4.1.6 Optimization

In adapting neural network, firstly I didn't normalize the data. Then in later calculation, I find that although we use the tf-idf model, but the difference between features cannot be fully used. And some words appear more frequently will be more powerful than less frequently ones. So I normalized the feature. And in this process I find that only normalize on tf data can achieve the same result. This finding help me reduce the idf calculate step. And in ANN, the most parameters are hidden layer scale and BP step length. Firstly, I only set 500 hidden layer nodes to simplify the calculation. But I find that the model doesn't converge. And then I add the node number from 500 to 1000, 2000. And the step length I set to be ***(dy/dx)/(10T)***. And this model can converge. And I find this method very tricky, the performance is highly uncertain. And after the error rate below 20%, the model usually come into overfitting on validation data. Which let me set the stop line as 20%

### 6.4.2 Advantage and Shortcoming:

Advantage is that it can find hidden relationship among features. But shortcoming is Slow. Since it use gradient, it doesn't promise for global minimization.

## 6.5 Nonlinear SVM Classifier

SVM is a supervised learning model which aims to minimize the wrong prediction number. It is usually used on two type classification task. In our task, we use SVM to determine whether the Text is belong to appointed category. The original algorithm provided on class was a linear classifier. In our idea, the task of text categorization is a nonlinear problem. So we try nonlinear SVM to do this work by applying kernel trick.

### 6.5.1 Dual form:

For support vector machine, weight is combination of samples:

$$\boldsymbol{w = \sum_{i=1}^{n} \alpha_i y_i x_i}$$

And our optimize problem can transform to:

$$\boldsymbol{L(\alpha) = \sum_{i=1}^{n} \alpha_i y_i x_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_i y_i y_j x_i^T x_i}$$

$$\boldsymbol{= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_i y_i y_j k(x_i\ x_j)}$$

For this form, ***k(xi, xj)*** is called kernel function. In linear SVM, it is product of x, but in non-linear SVM, we can use other function replace it.

Gaussian kernel: In machine learning, Gaussian function is popular kernel function used in SVM classifier.

Ad: Optimal the error rate. Classification is more accurate.

Shortcoming: Time consuming

# 7. RESULTS & ANALYSIS

## 7.1 Evaluation Criteria

As we stated in Section 3. In order to evaluate the accuracies of different models we use, we use four main criteria to evaluate the models. These four criteria are: **FPR** (False Positive Rate), **FNR** (False Negative Rate), **BER** (Balanced Error Rate) and **ACC** (Accuracy).

$$\mathbf{FPR = False\ positives\ /\ \#labeled\ positive}$$

$$\mathbf{FNR = False\ negatives\ /\ \#labeled\ negatives}$$

$$\mathbf{BER} = \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{FPR + FNR})$$

$$\mathbf{ACC} = \frac{\mathbf{True\ positives + True\ negatives}}{\mathbf{\#labeled\ negatives}}$$

## 7.2 Baseline

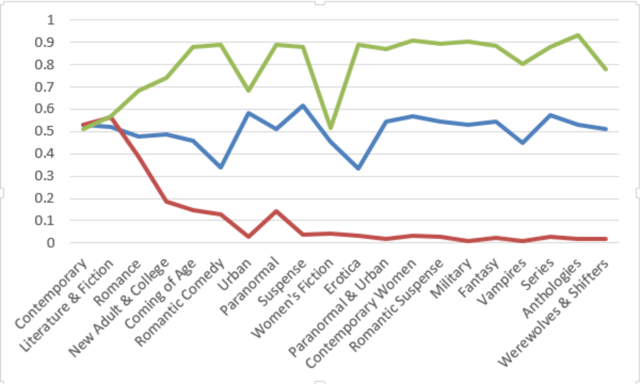

**Figure 2. Performance of baseline.**

The blue line is the accuracy line for 20 categories. Red line is FPR and green line is FNR.

We can see from the figure above that, the accuracy is about 50% while the FPR is low and FNR is high. It implies that the baseline kind of more not likely to categorize.

## 7.3 Results

**Table 3. The Decision Tree performance on 20 categories**

| Category No. | Accuracy | FPR | FNR | Labeled positive |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | 0.5810 | 0.3315 | 0.5121 | 4847(out of 10000) |
| 2 | 0.5455 | 0.5636 | 0.3678 | 5573 |
| 3 | 0.6389 | 0.1856 | 0.7836 | 2935 |
| 4 | 0.8033 | 0.0851 | 0.8425 | 1473 |
| 5 | 0.8709 | 0.0493 | 0.8784 | 962 |
| 6 | 0.8953 | 0.0306 | 0.8888 | 863 |
| 7 | 0.9269 | 0.0296 | 0.6852 | 664 |
| 8 | 0.8712 | 0.0415 | 0.6887 | 1349 |
| 9 | 0.8573 | 0.0394 | 0.8775 | 1233 |
| 10 | 0.8700 | 0.0430 | 0.9148 | 998 |
| 11 | 0.8897 | 0.0345 | 0.8905 | 886 |
| 12 | 0.9279 | 0.0200 | 0.8723 | 611 |
| 13 | 0.8829 | 0.0341 | 0.9102 | 947 |
| 14 | 0.8949 | 0.0268 | 0.8917 | 905 |
| 15 | 0.9518 | 0.0111 | 0.9017 | 417 |
| 16 | 0.9251 | 0.0219 | 0.8844 | 614 |
| 17 | 0.9607 | 0.0097 | 0.8043 | 373 |
| 18 | 0.9223 | 0.0278 | 0.8813 | 584 |
| 19 | 0.9378 | 0.0195 | 0.9336 | 467 |
| 20 | 0.9406 | 0.0198 | 0.7808 | 520 |

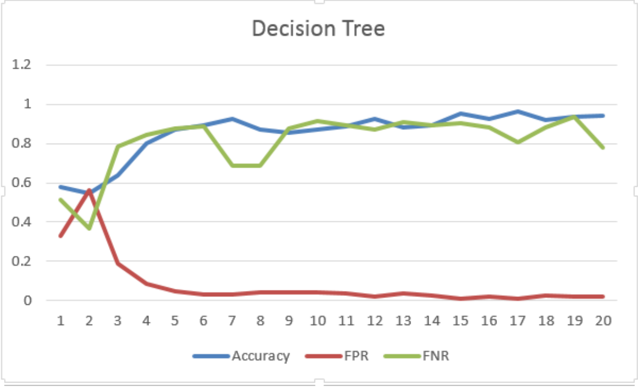

**Figure 3. The Decision Tree performance on 20 categories**

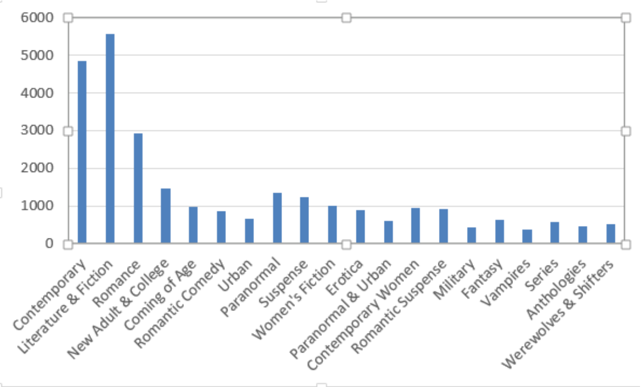

**Figure 4. The Decision Tree performance on 20 categories**

Table 3 and Figure 3, 4 are the examples of the results we collected.

**Table 4. Performance of different method (Category 1)**

| Method | Accuracy | FPR | FNR |
|---|---|---|---|
| Decision Tree/3000 features | 0.5734 | 0.4161 | 0.4318 |
| Decision Tree/selected features | 0.5810 | 0.3315 | 0.5121 |
| SVM | 0.6120 | 0.3127 | 0.4680 |
| Neural Network | 0.5757 | 0.3556 | 0.4973 |
| Linear Regression | 0.5138 | 0.4583 | 0.6353 |
| KNN | 0.4884 | 0.5357 | 0.5897 |
| Baseline | 0.4323 | 0.6456 | 0.6757 |

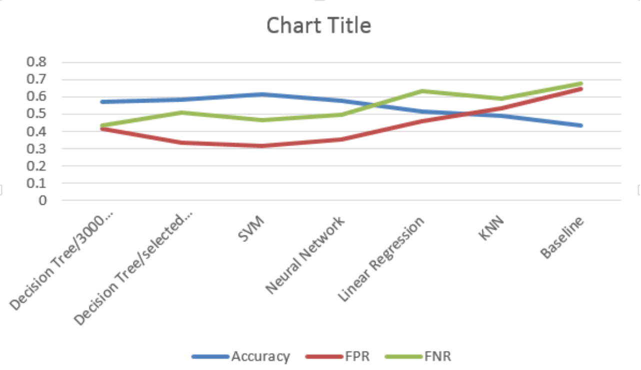

**Figure 4. Performance of different methods**

## 7.4 Analysis & conclusion

In first table, we can see that the accuracy raise as the difference between positive and negative sample number. At the same time, FPR decrease and FNR raise. But for all categories, the balanced error rate seems more stable. So we think BER can better fit the evaluation task. For balanced dataset, we can simply use accuracy to evaluate the performance of one classifier. But if there are far more positive/negative data than the opposite, only look at

accuracy will cause overestimation. For example, if only 1 percent of data is labeled positive, then predict all of them to be all negative will get a quite high accuracy. But in fact, this method get only 50% balanced error rate. And it cannot help us find probably positive sample at all.
And from the performance between raw feature and selected features, performance improves while features are reduced. From this point we know that select more feature sometimes will make classifier become worse. Then we examine the train error rate, find that the 3000 feature dataset has 95% accuracy, and the compressed features has 78% accuracy. We think this situation is surely overfitting. And by feature selecting and compressing, we can significantly mitigate overfitting.

Among all the method, SVM seems perform best. But at the same time, this method is also time-consuming. For example, train a decision tree using these data only cost nearly 1 minute. But training an SVM needs much more time, nearly half an hour. Same issue appear in NN method. From this comparison, we think the choice text mining algorithm should also consider the speed. Despite of some application really need fast response, low speed will at least cause difficulty on parameter tuning and method elaboration.

## 8. REFERENCES

[1] F Sebastiani. 2002. *Machine learning in automated text categorization*. ACM computing surveys (CSUR).

[2] Leo Breiman, Jerome H. Friedman, Richard A. O1-shen, and Charles J. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, l993

[3] Loh, Wei‐Yin. *"Fifty years of classification and regression trees."* International Statistical Review 82.3 (2014): 329-348.

[4] Reuters-21578 Text Categorization Collection. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[5] Tf-idf dataset(Matlab format). http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

[6] Chi-square definition.

http://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.htm

[7] State-of-the-art methods: Random Forest. Wiki: https://en.wikipedia.org/wiki/Random_forest

[8] Deep learning: http://deeplearning.net/

[9] LDA in 5 steps.

http://sebastianraschka.com/Articles/2014_python_lda.html#lda-in-5-steps

Text Analytics is the process of converting unstructured text data into meaningful data for analysis, to measure customer opinions, product reviews, feedback, to provide search facility, sentimental analysis and entity modeling to support fact based decision making. Text analysis software uses many linguistic, statistical, and machine learning techniques. Text analytics software allows users to gain insights from structured and unstructured data. The software mines text and uses natural language processing (NLP) algorithms to derive meaning from huge volumes of text. Organizations receive huge We review the related works that are about market prediction based on online-text-mining and produce a picture of the generic components that they all have. We, furthermore, compare each system with the rest and identify their main differentiating factors. Our comparative anal-ysis of the systems expands onto the theoretical and technical foundations behind each.Â  However, an analysis of emotional sentiment of news text can also be explored for market prediction. Schumaker, Zhang, Huang, and Chen (2012) have tried to evaluate the sentiment in financial news articles in relation to the stock market in his research but has not been completely successful. Keywords Translation based text categorization, Text categorization, Support vector machines, Arabic, Chinese, and Portuguese text categorization. I. INTRODUCTION The use of digital documents has widely increased over the years thanks to popularity of the Internet. Nevertheless, retrieving data and information from large numbers of documents is a challenging problem.