Adapting Feature Selection Methods for multi-class classification task

Bipra De
Indiana University
Bloomington,IN,USA
bde@indiana.edu

Nayana Charwad
Indiana University
Bloomington,IN,USA
ncharwad@indiana.edu

Sowmya Achanta
Indiana University
Bloomington,IN,USA
omachant@indiana.edu

Abstract:

Feature selection is an important step in sentiment analysis. A good set of features will go a long way in giving accurate results. In this project we shall discuss two approaches of obtaining features from a dataset: Information Gain (IG) and TF-IDF, and compare the results of both the approaches to determine which one fares better in terms of accuracy. We use user reviews from Epicurious, an informational website on food and cookery, to test both these approaches. In addition we also implement one vs all classification strategy for feature generation as opposed to binary classification of just good vs bad features. We use support vector machines (SVM-multiclass) to do the predictions on test data. In conclusion, we will discuss the results of our experiments.

1. Introduction:

Research on Sentiment Analysis gained importance in recent years (Pang and Lee, 2008; Bollen et al., 2011; Liu, 2012)[3]. Sentiment Analysis is the task of being able to analyze the polarity of a text/document e.g., good, bad, neutral or multi-level classes as in 1,2,3,4 star ratings for a movie. The words that occur in the document form the basis of such an analysis. These words can be associated with a set of features that can help us determine this polarity. Hence, feature generation/selection assumes great importance in such a scenario. To be able to bring about the correct set of features for each class of polarity, is in itself an important measure of the accuracy of our predictions about the sentiment of the document. There are many statistical methods in place to do the sentiment analysis. In this project, we've used the bigrams approach to generate features and support vector machine, SVM-multiclass (trained on a set of data vectors) to predict the ratings for test data. For this project, we consider a balanced dataset from Epicurious, an online community where recipes can be exchanged and reviewed. We investigate the behavior of the feature selection methods namely Information Gain and TF-IDF to predict the user ratings.

In the sections ahead, we describe: the two feature selection methods in detail, the data pre-processing steps, data-partitioning steps, feature selection steps (one vs all classification of training data in obtaining the relevant features), data vector generation, training and testing on SVM multi-class, evaluation steps. We conclude by showing the results obtained in both the approaches and the future scope of this project vis-a-vis cross fold validations, multiple feature selection methods, n-gram extensions.

2. Feature Selection:

Feature Selection is a process to extract relevant and useful features from the input data, to construct a model. This model can then be used to make useful predictions on new data based on the generated features. Usually, data may have information that is not of much use and it pays off to ignore such details in predictive models. Feature selection techniques aim at removing the redundant and irrelevant information from the data. The main advantage of feature selection is that it enhances the predictability factor of new data since irrelevant features will no longer affect the

results. The objective of feature selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data[1]. Feature selection is the subset of a two-step process:

1. **Feature Extraction:** This step ensures all the features are created from the input data and a score for each feature is assigned. It is based on this score that the next step vis-à-vis feature selection is done.
2. **Feature Selection:** Of all the generated features, only the top selected number of features will be chosen to be used as classification measure for the new data.

There are two types of feature selection methods: wrapper methods and filter methods. Filter methods attempt to assess the merits of features from the data, ignoring the effects of the selected feature subset on the performance of the learning algorithm, while wrapper methods assess subsets of variables according to their usefulness to a given predictor [2]. In this project, we've used filter methods because of their ability to deal with high-dimensional data, independence from the classification algorithm and being relatively simple, fast than the wrapper methods, which though take into account feature dependencies, suffer from the problem of over-fitting and involve intense computational cost.

In this project, we use the following approaches to extract features in the feature selection process. We then go on to compare the results from both the approaches.

For the discussion of the feature selection methods, we use the following notations[3]:

- S : target or positive class.
- S': negative class.
- D_s: The number of documents in class S.
- $D_{s'}$: The number of documents in class S'.
- D_{Sf} : The number of documents in class S where feature f occurs.
- $D_{S'f}$: The number of documents in class S' where feature f occurs.
- T_{Sf} : The number of times feature f occurs in class S.

2.1 Information Gain:

It is defined as the entropy of one random variable minus the conditional entropy of the observed variable. Thus, IG is the reduced uncertainty of class S given a feature f[3]:

$$IG = H(S) - H(S|f) = \sum_{f\in\{0,1\}} \sum_{S\in\{0,1\}} P(f,S) log \frac{P(f,S)}{P(f)P(S)}$$

Information gain is a good measure to decide the relevance of an attribute but can be problematic when there are large number of distinct values in the data. For this project, we've done one vs all classification of training data to generate features. For example: All 1-fork rating recipes are taken into one class and the 2,3,4-fork rating recipes to another class for the first run of features. Then all 2-fork rating recipes are considered into one class, and the rest into another, for the next run of features. The same is iterated for all the forks and the features obtained in each run are combined to form the final feature set that will be used in model construction (training the SVM).

2.2 TF *IDF:

TF–IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus[4]. Features are extracted using TF-IDF with the help of Lucene(*a free open source information retrieval software library, originally written in Java by Doug Cutting*) index, which is created on training data with 2 fields - "fork rating" and "reviews". The "fork rating" field has values 1-4 and the "reviews" field contains the concatenated string of user reviews for all recipes with a given fork rating (1-4). The top N features of reviews

and tips under each distinct category are calculated based on the TF-IDF score by the formula below.

$TF\text{-}IDF\ (fi) = [TF\ (f_i)]^2 * [\log10\ (|D|/D_f\ (f_i))]$

Where,

- f_i: Term
- TF (f_i): Frequency of a term f_i in a field of the Lucene document
- |D|: Total number of documents in the Lucene index
- D_f (f_i): Number of documents in the Lucene where the term f_i appears

2.3 Classifier:

We have used SVM multi-class[6] classifier (*SVM*^{multiclass} (Joachims, 2008)) with the default linear kernel. *[1]* For a training set (x_1,y_1) ... (x_n,y_n) with feature vector x_j and labels y_i in [1..k] (in our case the fork ratings 1 fork-4 fork) , it finds the solution of the following optimization problem during training.

$\min 1/2\ \Sigma_{i=1..k}\ w_i * w_i + C/n\ \Sigma_{i=1..n}\ \xi i$

s.t. for all y in [1..k]: $[\ x1 \bullet w_{yi}\ ] >= [\ x_1 \bullet w_y\ ] + 100*\Delta(y_1,y) - \xi 1$

.....

for all y in [1..k]: $[\ x_n \bullet w_{yn}\ ] >= [\ x_n \bullet w_y\ ] + 100*\Delta(y_n,y) - \xi n$

C is the usual regularization parameter that trades off margin size and training error. $\Delta(y_n,y)$ is the loss function that returns 0 if y_n equals y, and 1 otherwise.

3. Epicurious Data Set

For this project we have used data set of cooking recipes from Epicurious website[8]. On the website a recipe can be assigned a rating of 1 to 4 forks, with 1 fork being the worst and 4 fork being the best. There is also provision for ratings 1.5, 2.5 and 3.5. The ratings are accumulated over all user reviews. For experiment purpose we rounded down all half ratings so rating 1.5 will be considered as 1, rating 2.5 will be considered as 2 and rating 3.5 will be considered as 3.

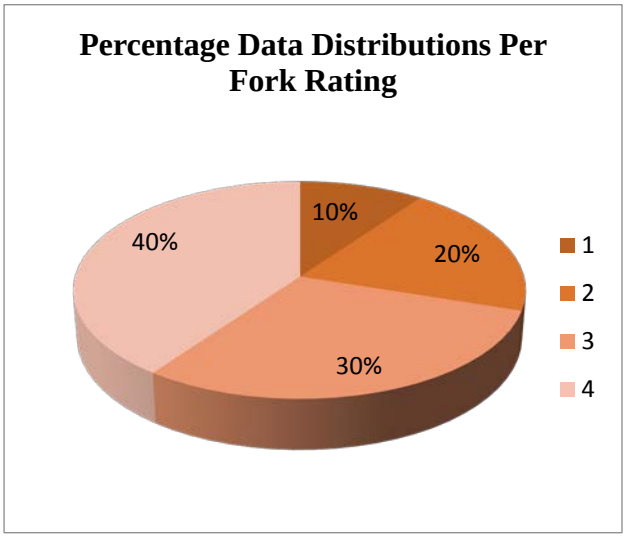

Figure 1. Percentage Data Distributions Per Fork Rating

We have total 26765 recipe reviews and their corresponding accumulated ratings. Based on all reviews, skewing ration can be calculated as follows:

Skewing Ratio = Number of recipes in major class / Number of recipes in minor class

Class C1 = Number of recipes in fork 3 and fork 4 (16715 + 7597 = 24312). Class C2 = Number of recipes in fork 1 and fork 2 (298 + 2155 = 2453).

Skewing Ration = 24313 / 2453 = 1 : 9.9

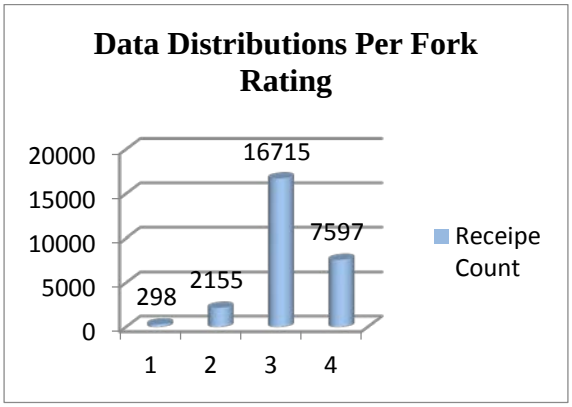

Figure 2. Data Distributions Per Fork Rating

For this project we randomly sampled total data set in two sets, training data set and test data set. Our training data set contain 60% of total data and our test data set contains 40% of total data.

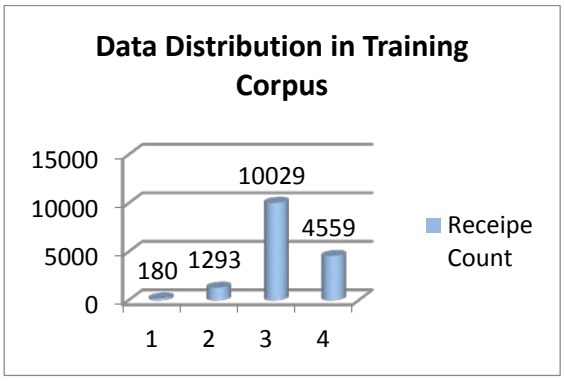

Figure 3. Data Distribution in Training Corpus

Since skewing ratio is considerably high, SVM may get biased towards positive recipe ratings (Fork 3 and Fork 4). So in order to create an unbiased SVM model we divided total number of recipes for each fork separately in training and test data set. For example, all recipes having 1-fork ratings are divided into 60% for training data and 40% for test data.

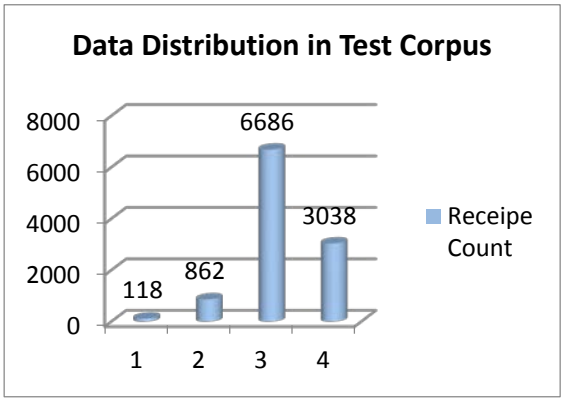

Figure 4. Data Distribution in Test Corpus

Next section of the paper will present different project tasks performed.

3.1 Project Tasks

Overall, our experiment can be divided into the following tasks that are further elaborated in next sections of the paper.

1. Data Preprocessing
2. Data Partitions
3. Feature Selection and Data Vector Generation
4. Training Support Vector Machine on training data vector
5. Classify test data with Support Vector Machine
6. Evaluate the results

3.1.1 Data Preprocessing

Data preprocessing is very important step to fine-tune the recipe reviews data in order to be acceptable by information gain (IG) and TF-IDF modules and also to filter noisy data. Following data preprocessing steps are implemented before the supervised machine-learning step.

Review File Preprocessing: For IG module, each recipe is represented as a list of comments, each comment is represented as a list of sentences and each sentence is represented as a list of words.

```
[
        [
        ["I","loved","the","recipe"],
        ["Very","Delicious","and","tasty"]
        ], ##Comment1
        [
        ["Difficult","to","make"]
        ], ##Comment2.
]
```

Then each preprocessed recipe file is converted into python pickle for further processing.

Stop Words Removal: For TF-IDF module, input recipe files are preprocessed to remove stop words so as to develop most appropriate features in feature selection module.

Case Conversion: All features are converted into lowercase so as to avoid getting duplicate

features which are present in both uppercase and lowercase.

3.1.2 Data Partition:

UNIX script is implemented to generate the list of file names present in training and test data sets and further to generate a list of file names for each fork rating. Then each file is converted into python pickle for further processing.

Feature Frequency: Features that occur less than 4 times in a given data partition are directly omitted from feature selection module on the assumption that less frequent features will be rarely present in test data sets and will not contribute much in rating prediction method.

Data Partition: Training data is further partitioned into following set of partitions:

- 2,3,4 fork ratings (Partition1) and 1-fork ratings (Partition2).
- 1,3,4 fork ratings (Partition1) and 2 - fork ratings (Partition2).
- 1,2,4 fork ratings (Partition1) and 3 - fork ratings (Partition2).
- 1,2,3 fork ratings (Partition1) and 4 - fork ratings (Partition2).

3.1.3 Feature Selection and Data Vector Generation:

During each experimental run, equal numbers of features are initially extracted from 1 fork ratings, 2 fork ratings, 3 fork ratings and 4 fork ratings. Then all these features are merged together into a single feature file after removing the duplicate features that will be further used for data vector generation.

We hypothesize that this approach will give us better set of features as compared to positive(3-fork and 4-fork recipes) and negative(1-fork and 2-fork recipes) partition approach. In positive negative partition method, maximum features get selected from those fork ratings which have maximum number of recipe files and thus a biased classification model towards the majority class gets generated.

3.1.4 Training Support Vector Machine on training data vector:

In our experimentation, we are addressing multi-class (1 fork, 2 fork, 3 fork,4 fork) classification using SVM multi-class classifier. After the data vectors are generated for the training and test data partitions using the feature file , the training data vector is fed to SVM to generate a trained model. We have set the "cost or penalty constant (c)" of the SVM classifier as 0.1 [7]. This constant sets the relative importance of maximizing the margin of class separation and minimizing the amount of classification errors. Thus, the lower this constant is, more generalized will be the trained model produced by SVM classifier, minimizing overfitting.

3.1.5 Classify Test Data with Support Vector Machines:

The trained model generated by the SVM multi-class classifier is then used to classify the recipes present in the test data set. The output file generated after classification has one predicted class (in our case it ϵ {1-fork, 2-fork,3-fork and 4-fork ratings}) per line. Each line in this output file denotes the fork rating predicted for a recipe file in the order in which the recipe files are present in the test data vector.

3.1.6 Evaluate results:

The class or fork predictions in the output file generated by the trained SVM classification model is compared against the ground truth i.e. the average ratings present in the recipe file names in the test data partition. For performance measure, precision, recall and accuracy is computed separately for each fork ratings.

For a given fork rating J,

Precision *= Number of recipes in the test data set that are correctly classified as J by the classifier / Total number of recipes in the test data set that are classified as J*

Recall *= Number of recipes in the test data set that are correctly classified as J by the classifier / Total number of recipes in the test data set with*

fork rating as J

Accuracy *= Total number of recipes that are correctly classified by the classifier / Total number of recipes present in the test data set*

Precision / Recall per fork rating

The following graphs denote the results obtained using Information Gain approach for feature selection. We varied the number of features selected to prepare the data vector in each run of our experiment and plotted the results.

1-Fork

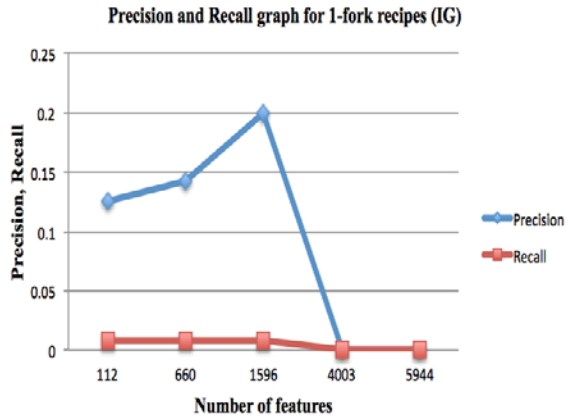

Figure 5. Precision and Recall graph for 1-fork recipes(IG)

2-Fork

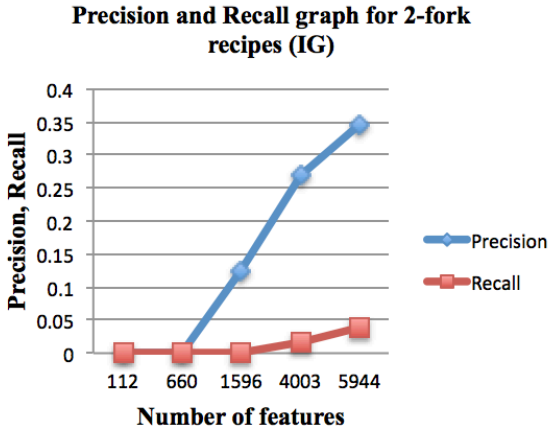

Figure 6. Precision and Recall graph for 2-fork recipes(IG)

3-Fork

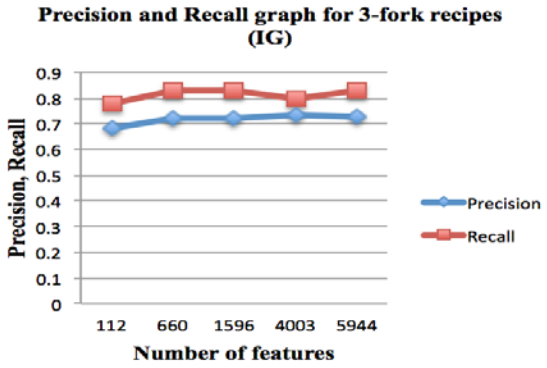

Figure 7. Precision and Recall graph for 3-fork recipes(IG)

4-Fork

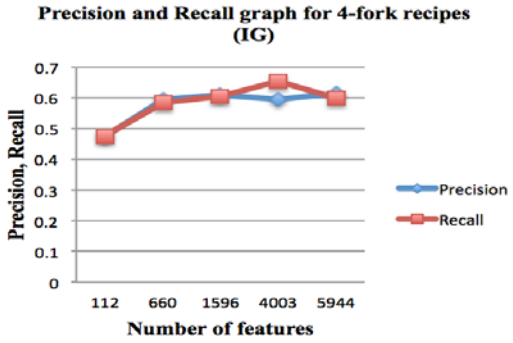

Figure 8. Precision and Recall graph for 4-fork recipes(IG)

The following graphs denote the results obtained using TF-IDF approach for feature selection. We varied the number of features selected to prepare the data vector in each run of our experiment and plotted the results.

1-Fork

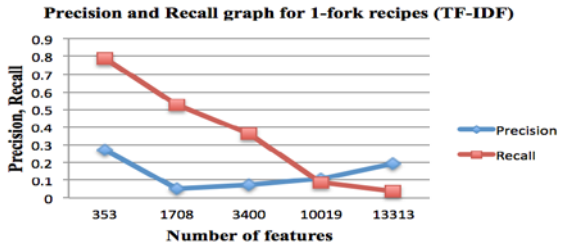

Figure 9. Precision and Recall graph for 1-fork recipes(TF-IDF)

2- Fork

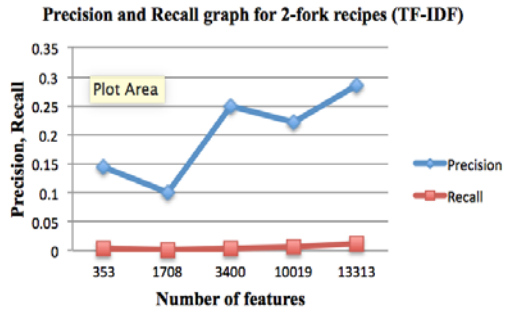

Figure 10. Precision and Recall graph for 2-fork recipes(TF-IDF)

3-Fork

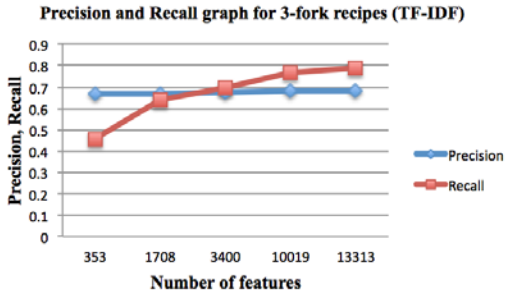

Figure 11. Precision and Recall graph for 3-fork recipes(TF-IDF)

4-Fork

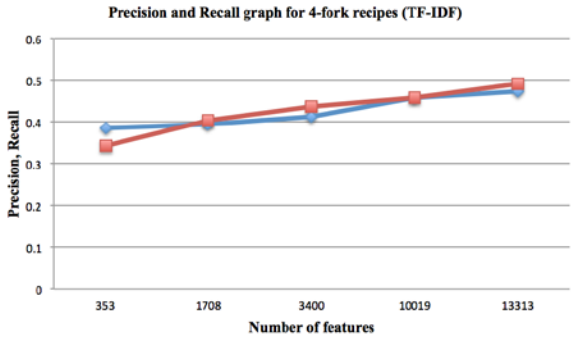

Figure 12. Precision and Recall graph for 4-fork recipes(TF-IDF)

Results:

Positive-Negative Multiclass Approach

Training Data set was partitioned into two sets - POS (1-fork and 2-fork recipes) and NEG (3-fork and 4-fork recipes)- with skewing ratio of 1:9.9. Information Gain is used as the feature selection method. Then, after training the SVM multi-class classifier and running it against the test data, we observed that SVM binary classifier outperforms SVM multi-class classifier in the similar setup in terms of accuracy. Based on results obtained in paper [3], for skewing ratio 1:9.9 average accuracy for SVM binary classifier should be close to 88.9%. But, the SVM multi-class classifier with similar setup and skewing ratio gives an average accuracy of only 66.6%. Below graph represents accuracy achieved for positive-negative partition based multiclass approach.

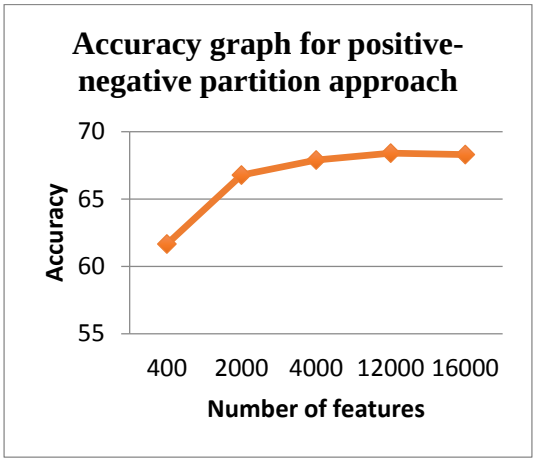

Figure 13. Accuracy graph for positive-negative partition approach

One Vs All Multiclass Approach For Feature Selection

Selecting features (using Information Gain) from the majority classes (3-fork and 4-fork recipes) proved to be less efficient in terms of accuracy when compared to feature selection from all the classes i.e. both minor (1-fork and 2-fork recipes) and major (3-fork and 4-fork recipes). In latter case, higher accuracy is achieved with almost half number of features when compared to the former approach of feature selection. Below graph represents the accuracies achieved for Information Gain approach.

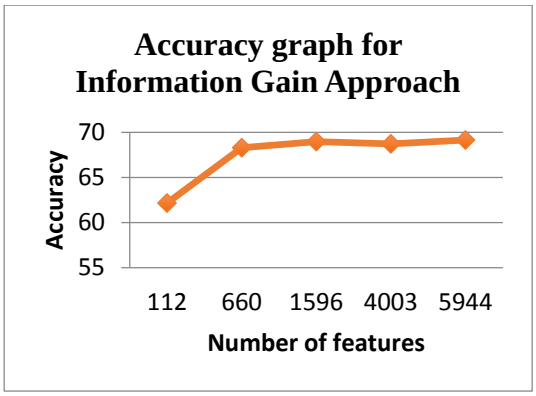

Figure 14. Accuracy graph for Information Gain approach

Following graph represents the accuracies achieved for TF-IDF approach.

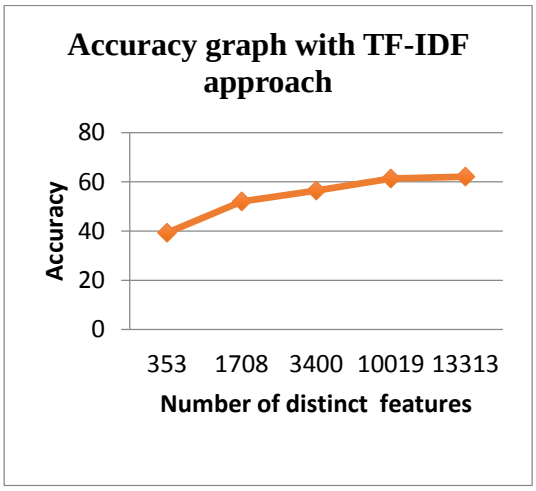

Figure 15. Accuracy graph for TF-IDF approach

As per Figure 14 and Figure 15, Information Gain is a better approach for feature selection than TF-IDF because with less than half number of features better accuracy is achieved for all our experimental runs in the Information Gain approach.

4. Conclusion and Future Work:

In this paper, we compared the performance of the two feature selection methods — Information Gain and TF * IDF — for a multi-class classification problem and found out that Information Gain performs significantly better in our experiment setup. We also observed that, in a highly skewed data set, selecting features from all classes (both major and minor) leads to an improvement in accuracy when compared to selecting features from only majority class to train SVM.

For future work, multi stage classification can be implemented wherein we first build a binary classifier on positive (3-fork and 4-fork recipes) and negative (1-fork and 2-fork recipes) data partitions of the training set. Once a recipe is classified using the binary classifier into one of these partitions, it can be subjected to further classification under the positive and negative partitions to classify it finally into a single class or fork rating.

N-Fold Cross validation can be implemented to give a better generic estimate of the accuracy of the classifier when subjected to unseen test data for classification.

Our current experimental setup on bigram features can be further extended to measure the performance of the classifier with unigram and trigram features.

References:

1. http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf
2. http://www.ulb.ac.be/di/map/gbonte/bioinfo/course4.pdf
3. Feature Selection for Highly Skewed Sentiment Analysis Tasks, Can Liu, Sandra Kuebler, Ning Yu
4. Rajaraman,A.;Ullman,J.D(2011)."MiningofMassiveDatasets".pp.1to17. doi:10.1017/CBO9781139058452.002. ISBN 978113905842
5. www.wikipedia.org
6. http://svmlight.joachims.org/svm_multiclass.html
7. http://www.brainvoyager.com/bvqx/doc/UsersGuide/MVPA/SupportVectorMachinesSVMs.html
8. www.epicurious.com