1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT 2019)

Topic Classification from Text Using Decision Tree, K-NN and Multinomial Naïve Bayes

Md. Ataur Rahman
Dept. of Language Science and Technology
University of Saarland
Saarbrücken, Germany
arahman@coli.uni-saarland.de

Yeasmin Ara Akter
Dept. of Computer Science and Engineering
East Delta University
Chittagong, Bangladesh
yeasmin.a@eastdelta.edu.bd

Abstract—**One of the central motivations behind Natural Language Processing is detecting patterns. Given a text document, the task of identifying the context is known to be as topic classification. This paper explores the performance of three different classifiers namely Decision Tree, K-Nearest Neighbors and Multinomial Naive Bayes on a topic classification task (with six topic class). The evaluation is done on the basis of accuracy, precision, recall, and f1-score based results. Among those three aforementioned classifiers, we have selected the Multinomial Naïve Bayes as our best model using which we achieved 91.8% accuracy.**

Keywords—**Text Classification, Natural Language Processing, Machine Learning**

I. Introduction

Topic classification (TC) is one of the classic subdomains of natural language processing which deals with allocating predefined classes to text documents. Due to the abundance of textual information in digitalized format, the need for categorization of documents has become a thriving interest for the past few decades. The first idea probably originated back in 1961 from the seminal work of Maron on probabilistic text classification [1]. One dominant approach to tackle this problem is based on machine learning techniques, which recently took the research communities by storm.

The prime inquisitiveness in most of the text classification task spans from finding out the best possible combinations of features with best Machine Learning (ML) classifiers. Our motivation in this article also extends towards the investigation of how different supervised machine learning classifier performs on a multi-label text classification task. We tried to experiment with the parameters of three different classifiers (Decision Tree, K-NN and Multinomial Naive Bayes) implemented using the scikit-learn library. We managed to elevate the performance of each of these baseline classifiers by employing some form of pre-processing (stemming, stop-word removing, lemmatization etc.). Apart from that, both *tf-idf* and *count* features were explored in order to compare the results for the aforementioned classifiers under different settings.

II. Related Works

Perhaps one of the earlier prominent work in this domain is the usage of Support Vector Machine (SVM) for learning to classify text [2]. The paper delineates a comparison between SVM and four traditional machine learning approaches. Two corpora were used for evaluation and an SVM with an RBF-kernel gained 86.4% and 66.0% accuracy respectively.

This paper [3] describes the correction scheme of the Reuters Corpus Volume I (RCV1-v2) from the original RCV1-v1. Apart from this, they have used the above amended corpus to evaluate three popular ML algorithms (SVM, k-NN, Rocchio). A micro F1 score of 0.816 and macro F1 of 0.619 was obtained using an SVM.

A paper [4] by Sebastiani et al. provided an in-depth exploration of nearly every type of classification techniques including the neural network and could be recalled as a starting point in our research. The paper also compared state-of-art systems at that time among which the best-reported method [5] using SVM obtained an accuracy of 92% and Decision Tree (DT) gave 88.4% accurate results using the Reuters corpus.

Another article [6] provides a heuristic on character-level Convolutional Neural Networks (CNN) for the task of textual document classification. They have compared several traditional as well as deep learning models and claimed that character-level CNN is a better choice in terms of data scalability.

III. Dataset

For our experiment, we have considered a fraction of Amazon's product review corpus consisting of 6000 text documents from which 75% (4500) documents are used for *training* purpose. The rest 25% (1500) data are used for *testing* the performance of the system. Each review is annotated as one of *six* topic type categories (*books, camera, dvd, health, music, software*) which corresponds to multi-class variants. The distribution of classes in both training and testing data is shown in TABLE I.

TABLE I. Distribution Of Various Labels In The Dataset.

Labels	**Training Set**	**Testing Set**
Books	760	233
Camera	730	258
DVD	770	242
Health	743	243
Music	767	260
Software	730	264
Total	4500	1500

IV. Methodology Of The System

In this section, a brief idea about the settings and approach taken for all the three classifiers is given. But before going to the details about the classifiers we will have a look at the *pre-processors* used in our system:

A. Preprocessing

We have experimented with three types of preprocessing techniques that are popular in text analysis domain. First of all, we have removed the *stop words* (such as *a*, *of*, *the* etc.) which occurs regularly in English text and do not generally contribute that much in classification. The second pre-processing techniques that we followed is the *stemming* of multiple forms of a single word into the same root. We have used *Porter's Stemmer* from NLTK for stemming. Both of these improved the results substantially for example in the case of K-NN classifier. The accuracy improves from 75.7% to 82.6% (7% for tf-idf feature) and from 41.87% to 59.1% (17.23% for count feature). The other classifiers performance also improves between 1-2%.

Another tool that we experimented with our system is the WordNet *lemmatizer* from NLTK which didn't improve the performance, this may be due to the reason that, for topic classification task it is important to preserve the word as it is instead of converting it to lemma.

B. Decision Tree

Decision Tree (DT) constructs classification models by breaking down a dataset into smaller subsets while incrementally building an associated tree. The final tree contains *decision nodes* and *leaf nodes* (Fig.1). A decision node consists of two or more branches (e.g., color, size and shape) while a leaf node (e.g., edible) represents a decision or classification. The best predictor, also referred to as the **root node**, is marked by the topmost decision node. DT can operate on either numerical or categorical data. Perhaps the intuition for the decision tree can be best described by an example (Table II):

TABLE II. An Example Dataset.

Color	Size	Shape	Edible
Yellow	Small	Round	Yes
Yellow	Small	Round	No
Green	Small	Irregular	Yes
Green	Large	Irregular	No
Yellow	Large	Round	Yes
Yellow	Small	Round	Yes
Yellow	Small	Round	Yes
Yellow	Small	Round	Yes
Green	Small	Round	No
Yellow	Large	Round	No
Yellow	Large	Round	Yes
Yellow	Large	Round	No
Yellow	Large	Round	No
Yellow	Large	Round	No
Yellow	Small	Irregular	Yes
Yellow	Large	Irregular	Yes

A decision tree utilizes *entropy* to compute the homogeneity from a sample. *Information gain is* another terminology which indicates the mitigation of entropy when a dataset is partitioned based on an attribute. Constructing a node in the DT thus involves finding attribute that produces the maximum *information gain*. Thus, the first Branch of the tree in Fig.1 consist of the attribute *size* because it has the maximum Information Gain ($0{:}106$) among all the three features [$G(color) = 0{:}036$; $G(shape) = 0{:}036$].

In the scikit-learn implementation of the Decision Tree classifier, we have experimented with several parameters such as *maximum depth* of the tree, *minimum samples* to split an internal node, *minimum samples* per leaf node, *maximum number of features* and etc. We have tried different

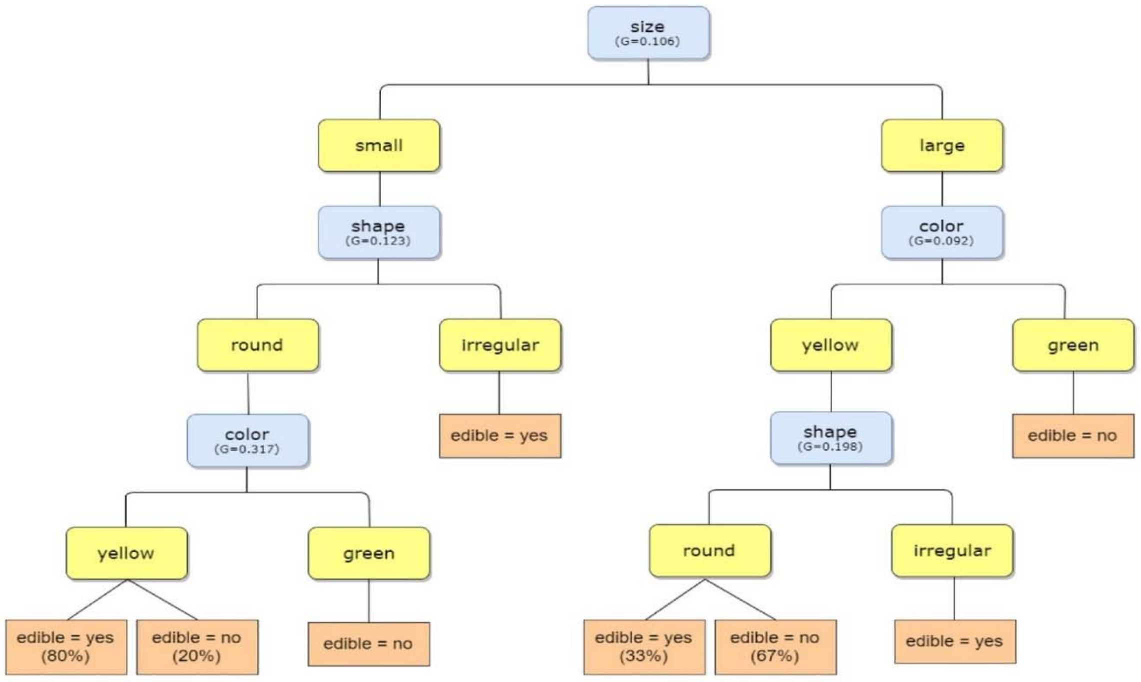

Fig.1. Decision Tree for the dataset of TABLE II.

combinations by changing these parameters. One such setting for example, is given below:

```
  DecisionTreeClassifier
 (min_samples_split=3,
min_samples_leaf=2, max_depth=10,
max_features=1000)
```

Changing the value of parameter reduces the performance of the classifier from an accuracy of 79.4% (tf-idf feature) and 79.93% (count feature) to 48.47% and 47.4%.

C. K-Nearest Neighbors

The second classifier that we compared for performing a six-way classification was the K-NN classifier. By experimenting with different values of *K*, we have observed the following phenomenon after plotting the *accuracy* and *f1-score* (Fig.2.). The observations are as follows:

a) Accuracy gets better with a higher K in the observed experiment.

b) The overall performance (both accuracy and f1-score) increases drastically from *K=1* to *K=10*. After that, the curves seem to have more like a plateaued progression having upward trends with some fluctuations.

c) The *bias* for K-NN classifier increases monotonously with the value of *K*, while the variance drops off as *K* is increased. In fact, under reasonable assumptions, the bias of the 1-nearest neighbor (*K=1*) classifier disappears completely with the size of the training data approaching infinity.

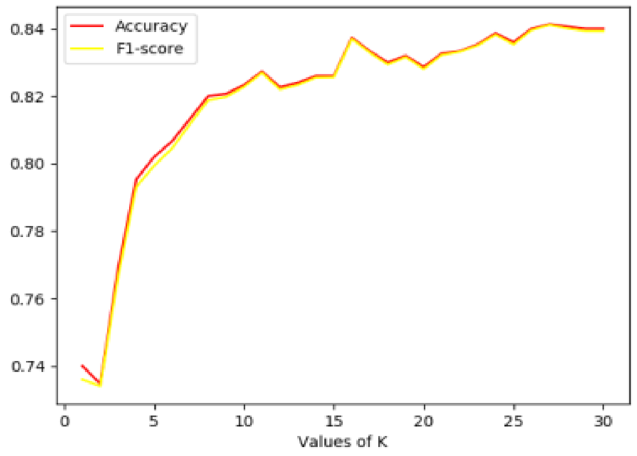

Fig.2. Performance of K-NN classifier for different values of K.

A comparison based on *time* (in seconds) taken for training and testing the other two classifiers with K-NN is given below in Table III.

TABLE III. COMPARISON OF TRAINING AND TESTING TIME.

	NB	**DT**	**KNN**
TRAINING(S)	0.906	3.914	0.789
TESTING(S)	0.215	0.208	1.015

From Table III, we can see that the different classifier has different training and testing time. The lowest training time is achieved by K-NN classifier (0.789) while the highest training time is for Decision Tree (3.914). The testing time, on the other hand, is highest for K-NN (1.015). These training and testing time heavily depend on the architecture of the classifier itself. As building tree takes linear time *O*(*n*) with a searching cost of *O*(log *n*), the overall complexity gets high for DT. For Naïve Bayes it's fairly simple, it takes only the time to construct the frequency table and the multiplication time to calculate the conditional probability of the individual words. On the contrary, K-NN has properties that are quite different from most other classification algorithms. Training a K-NN classifier simply involves the task of determining *K* and preprocessing documents. In fact, if we pre-select a value for *K* and do not pre-process, then K-NN requires no training at all. For *n* samples in *D* dimensions, the test time for k-NN is *O*(D n log *n*). It is linear in the size of the training set as we need to compute the distance of each training document from the test document. Test time is independent of the number of classes. K-NN therefore, has a potential advantage for problems with large class size.

D. Multinomial Naïve Bayes

The third classifier that we have used is the "Multinomial Naive Bayes" classifier which is a slightly modified version of the Naive Bayes classifier. The only difference is that while the Naive Bayes works with conditionally independent features the modified Multinomial version could be used to estimate the likelihood of conditionally dependent features by considering them as independent features. As it uses a bag of word feature in the text, it assumes that there is no relation between each word through the text, which is okay as long as we don't consider the semantic relationships of words in a sentence or among the sentence.

V. RESULTS AND EVALUATION

In this section, we will compare the results of three different classifiers mentioned in the previous section (Section III). For our evaluation purpose, we considered *accuracy*, *precision*, *recall* and *f1-score* based measures. We have also used a *10-fold cross validation* over the entire dataset for further evaluation regarding each type of classifiers performance.

For the purpose of simplification, we have only illustrated the comparison of the overall score (micro average) for the three classifiers (Table IV and V). Based on the comparison, we have chosen *Multinomial Naive Bayes* classifier with *Tf-Idf feature* as our *Best Model*. A detailed result for each of the class label for the best model is depicted in Table VI.

TABLE IV. COMPARISONS OF RESULTS FOR DIFFERENT CLASSIFIER (TF-IDF FEATURE).

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
DT	0.788	0.794	0.788	0.790
KNN	**0.826**	0.831	0.826	0.826
NB	**0.918**	0.920	0.918	0.918

TABLE V. COMPARISONS OF RESULTS FOR DIFFERENT CLASSIFIER (COUNT FEATURE).

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
DT	**0.796**	0.802	0.796	0.798
KNN	0.591	0.613	0.591	0.590
NB	**0.905**	0.908	0.905	0.905

TABLE VI. DETAILED RESULTS OF MULTINOMIAL NAÏVE BAYES CLASSIFIER.

	precision	**recall**	**F1-score**	**support**
Books	0.935	0.923	0.929	233
Camera	0.862	0.942	0.900	258
DVD	0.920	0.901	0.910	242
Health	0.971	0.840	0.901	243
Music	0.947	0.962	0.954	260
Software	0.892	0.936	0.913	264
Avg/Total	0.920	0.918	0.918	1500

To have a clearer look at the accuracy results, we have also evaluated our classifiers with the 10-fold cross-validation scheme. These are depicted in Table VII.

TABLE VII. 10-FOLD CROSS VALIDATION ACCURACY RESULTS FOR THREE DIFFERENT CLASSIFIERS(BASED ON BEST RESULTS).

Folds	**NB**	**DT**	**KNN**
Fold-1	0.926	0.797	0.856
Fold-2	0.863	0.778	0.810
Fold-3	0.882	0.750	0.796
Fold-4	0.893	0.800	0.793
Fold-5	0.886	0.731	0.825
Fold-6	0.812	0.731	0.731
Fold-7	0.792	0.772	0.778
Fold-8	0.926	0.832	0.852
Fold-9	0.872	0.757	0.750
Fold-10	0.858	0.723	0.777

VI. CONCLUSION

From the result section, we can see that the overall accuracy for Multinomial Naive Bayes classifier for the task of topic classification in our experiment was 91.8% which is better than Decision Tree (79.6%) and K-NN (82.6%) classifier at their best settings. This is the prime motivation behind selecting *Multinomial Naive Bayes* as our best model. Again, the time consumed by MNB (Table III) is a good trade-off in terms of both training and testing perspective. The approach that MNB takes is also straight forward to understand as it directly considers the prior and posterior probabilities for classification purpose and there is no further complexity under-the-hood. We have used three different pre-processing techniques in our experiment. Although, it would have been nice to incorporate other normalization and feature selection techniques to see if the result improves or not, nevertheless what we have achieved is pretty much similar to the state-of-the-art systems in this domain.

REFERENCES

[1] M. E. Maron, "Automatic indexing: an experimental inquiry," *Journal of the ACM (JACM),* vol. 8, pp. 404-417, 1961.

[2] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, 1998.

[3] D. D. Lewis, Y. Yang, T. G. Rose and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of machine learning research,* vol. 5, pp. 361-397, 2004.

[4] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR),* vol. 34, pp. 1-47, 2002.

[5] S. Dumais, J. Platt, D. Heckerman and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge management*, 1998.

[6] X. Zhang, J. Zhao and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015.