News Recommendation System Based on Multiple Classifiers

Vivek Pabani
Illinois Institute of Technology
vpabani@hawk.iit.edu

Anup Deulgaonkar
Illinois Institute of Technology
adeulgao@hawk.iit.edu

Abstract

In this project, we have designed, implemented and analyzed a news recommendation system. We used news articles dataset provided by BBC for the experiments. The system trains multiple classifiers on the training set, and use them to classify the news article chosen by the user. It then recommends m similar articles to the user. In this report, we discuss the methods we have implemented, results and their analysis. We have got best results with Naive Bayes algorithm, although we use combination of all the classifiers.

I. Introduction

With rapid increase in digital data, it is hard to find content of interest. Be it music, articles, products or books, there is vast amount of data online, and filtering it according to preference at user level is difficult. A recommendation system works in direction to solve this problem. It takes the user preference and user's history of choice in account, and recommends similar content which may interest the user.

We have implemented such a recommendation system for news articles. Almost all the online news service providers use one or other kind of recommendation system nowadays. The basic concept is the same - if we have history of user's previously read articles, we can use that information and decide if any new article will be of his interest or not. Based on this decisions, we can recommend new articles to the user. Our application is rather simple. We provide n article titles to the user, and based on his choice, we recommend m new articles, where n and m are user selected. The original n articles are picked from test dataset, where topics are unknown at the moment, and m recommendations are given from entire dataset. User can select the articles based on title, and can read the text by further options.

II. Dataset

We have used BBC news dataset for these experiments. the dataset consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004-2005. The topics are business, entertainment, politics, sport and tech. Each document has a title and article text. All three classifiers are trained and tested on the same training and testing dataset. The data is divided in 10 folds, where one part is used for testing and rest is used for training.

III. Algorithms

We have used three classifier algorithms - KMeans, Naive Bayes and Rank-Classifier for the classifying process. K-Nearest Neighbor is used for recommending K closest documents from the target document.

A. KMeans

K-means is a method of vector quantization, which is popular for clustering analysis. K-means clustering aims to partition N observations into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results is a partitioning of the data space into Voronoi cells.

K-means algorithm originally is a NP hard problem but using some intelligent heuristics can converge to a local optimum if we are working in non-convex curving function. K-means clustering uses Expectation Maximization approach which is an iterative approach to converge to local Minimum. K-means algorithm is implemented with cluster centroids with comparable special extent and we use hard clustering i.e. documents do not overlap into multiple clusters.

Algorithm Pseudocode:

- Pick K mean vectors using labeled data.
- Calculate initial mean and allow documents to assign to different cluster contradicting the label tags. We do this step to not over fit the data.
- Iterate until $|\mu_j^{new} - \mu_j^{old}| < \epsilon$:
  - Assign each document x_i to its closest mean vector μ_j.
  - Update each mean vector μ_j to be the mean of the x_i's assigned to it.

Distance between documents and mean are calculated using Cosine Similarity (since the documents are normalized according to their length).

An error function is used as Gradient descent and the objective is to minimize this error function. It is the sum of the distance between the documents to their assigned clusters.

$$E(D, M) = \sum_{i=1}^{N} \sum_{j=1}^{K} r_{ij} \cdot d(x_i, \mu_j)$$

B. Naive Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

The key is how to compute the posterior probability $P(c_i|\vec{d}_j)$ that document d_j belongs to category c_i. According to Bayes formula, the posterior probability $P(c_i|\vec{d}_j$ is translated to compute the prior probability $P(\vec{d}_j|c_i)$. Then, the categories that have the most prior probability are judged into the final categories of document d_j.

$$P(c_i|\vec{d}_j) = \frac{P(\vec{d}_j|c_i) \cdot P(c_i)}{P(\vec{d}_j)}$$

Here $P(c_i)$ denotes the probability of category c_i in the training set and $P(d_j)$ denotes document d_j in the training set. Because $P(d_j)$ is invariant for a given document d_j in all categories. The final category is decided by following formula :

$$argmax_{c_i} P(c_i|\vec{d_j}) = argmax_{c_i} P(\vec{d_j}|c_i) \cdot P(c_i)$$

We have used Multinomial Naive Bayes, in which we take into account the term frequency in the class, the term count of the class and vocabulary of the dataset. If $\vec{d_j} = \{w_1, w_2, ..., w_n\}$, then the probability of a token w_j given class c_i is calculated by

$$P(w_j|c_i) = \frac{count(w_j, c_i) + 1}{count(c) + |V|}$$

Using this, the probability of a document given the class is given by

$$P(d_j|c_i) = P(c_i) \cdot P(w_1|c_i) \cdot P(w_2|c_i) \cdot ... \cdot P(w_n|c_i)$$

C. Rank-Classifier

The rank-classifier focuses on a modified version of document indexing method tfidf, called tfidfie, where 'ie' stands for information entropy. This was introduced by Qirui Zhang in the paper 'Machine Learning Methods for Medical Text Categorization'. This indexing help remove those terms that do not affect the class distinction. Once the index is available, the target document is assigned a score calculated as the sum of tfidfie values of its words. The documents are ranked based on these scores, and the final class is assigned based on highest ranking.

The original and modified indexing methods are explained further.

C.1 Standard TFIDF

$tfidf$: Term Frequency - Inverse Document Frequency - is a standard weight function used for document indexing. This function takes in account the term frequency in the document, and the term distribution among the dataset. It is defined as :

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot log(\frac{|T_r|}{\#_{T_r}(t_k)})$$

Here t_k denotes a term, d_j denotes a document, T_r denotes the training set, $\#(t_k, d_j)$ denotes the term frequency in the j^{th} document, that is, the number of times t_k occurs in d_j, $\#|T_r|$ denotes the number of all documents in the training set, and $\#_{T_r}(t_k)$ denotes the document frequency, that is, the number of documents in T_r in which t_k occurs.

This function assigns the weight to a term in a document in a way that - if the term occurs in less documents, it gets more weight. In addition, id the term frequency in that document is high, it gets even more weight. On the other hand, if the term occurs frequently in different documents, it is a common term, and should not be given more importance for the classification. So, it gets low weight. Also, if the term frequency in that document is low, it gets even lower weight. Thus, it finds the term which are not common among the training set, and assigns high importance to them in the document.

Usually the weights are transformed into [0, 1] range for universal and cross document/method comparison. In order to do that, we perform cosine normalization on the $tfidf$ vectors.

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T_r|}(tfidf(t_s, d_j))^2}}$$

C.2 Modified TFIDFIE

Even though $tfidf$ is a standard function, $tfidf$ cannot be perfectly adapted to the inherent characteristic of text classification, because it only considers the term frequency and document frequency, and does not consider the distribution of those documents containing the term in the training set.

In text classification, the importance of term depends on not only its term frequency and document frequency, but also its contribution to classification. For example, term t_1 and t_2 have the same term frequency in document d_1 and the same document frequency in the training set, but the documents in which t_1 occurs only appear in one category and those documents in which t_2 occurs uniformly appear in each category. Obviously, t_1 is more important than t_2 for classification, i.e., t_1 should have higher weight than t_2. However, if we compute weights by the standard tfidf function, weight of t_1 is equal to t_2.

In order to solve this problem, we also calculate the distribution of those documents in categories in course of weighting terms. This distribution can be weighed by information entropy H, defined as :

$$H(t_k, d_j) = -\sum_{l=1}^{|C|} \frac{DF_{kl}}{\#_{T_r}(t_k)} \cdot log(\frac{DF_{kl}}{\#_{T_r}(t_k)})$$

Here $\#_{T_r}(t_k)$ denotes the document frequency, that is, the number of documents in T_r in which t_k occurs, DF_{kl} denotes the number of documents in category c_l in which t_k occurs, and $|C|$ denotes the number of categories in T_r.

Now, as per the maximum entropy theorem, if a term is uniformly distributed among the training dataset, it gets high entropy H value. The highest value of H occurs when a term occours in all the documents, while lowest value of H occurs when it's a unique term with respect to categories. We can use this property to modify $tfidf$ to make $tfidfie$, such that it assigns higher value to rare terms and low value to common terms among the categories.

$$tfidfie(t_k, d_j) = \frac{\#(t_k, d_j) \cdot log(\frac{|T_r|}{\#_{T_r}(t_k)})}{-\sum_{l=1}^{|C|} \frac{DF_{kl}}{\#_{T_r}(t_k)} \cdot log(\frac{DF_{kl}}{\#_{T_r}(t_k)})} = \frac{tfidf(t_k, d_j)}{-H(t_k, d_j)}$$

With this function, 1. if a term occurs in one category, it gets high weight. 2. if that term occurs more number of time in that document, it gets even higher weight. 3. if a term is uniformly distributed among all the categories, it gets low weight. 4. if that term occurs even low number of time, it gets lower weight. Thus, only terms that define the category well gets high weight in the end.

D. K-Nearest Neighbor

The K-Nearest Neighbors algorithm is a non-parametric method used for classification and regression. It has been widely used for text categorization.

The model for kNN is the entire training dataset. When a prediction is required for a unseen data instance, the kNN algorithm will search through the training dataset for the k-most similar

instances. The prediction attribute of the most similar instances is summarized and returned as the prediction for the unseen instance. The decision rule in kNN can be written as:

$$y(x, c_i) = \sum sim(x, d_j) \cdot y(d_j, c_i)$$

Here, $sim(x, d_j)$ is cosine similarity between two documents based on their vector representation. $y(x, c_i)$ returns the value 1 or 0 based on the category prediction for i^{th} category.

We have used K-Nearest Neighbor in the final stage of recommendation system in order to find the k closest articles from the source document. This is done after the other algorithms have correctly classified the document first.

IV. Experiments

A. Experiment Setting

As mentioned, we have used BBC dataset divided into five categories. Each article contains a title and text, and they are treated separately in different algorithms. The data is tokenized by removing punctuations, and stop words and small terms(length < 2) are removed as well. We used WordNetLemmatizer from nltk library on the data.

B. Multiple Classifier Decision and Recommendations

We have used partial Bootstrap aggregating, or Bagging algorithm to decide the final decision of class. The target document is classified with all three - KMeans, Naive Bayes and Rank Classifier. The final prediction is taken by the majority vote of three decisions. In case where all three classifier differ in their prediction, the prediction of the classifier with the highest F-Measure is considered as the final decision.

Once the decision is final, KNN algorithm is applied on the documents of the same class as the decision and K closest documents from the target document are returned as the recommended documents.

C. Performance Evaluation

The confusion matrix is created with self defined method. It depicts the total number of records for true and assigned class for all classes. We use recall (r), precision (p) and F1 measure to evaluate the classification. In our experiments, micro-averaging is used, i.e., these scores are computed globally over all the nm binary decisions where n is the number of total test documents, and m is the number of categories performance.

D. Result and Analysis

The results in the table are taken as average of 5 experiments done on random split of the dataset.

	K-Means	Rank-Classifier	Naive Bayes
Accuracy	0.7058	0.8982	0.9728
Precision	0.6765	0.8977	0.9715
Recall	0.832	0.8995	0.9723
F-Measure	0.7454	0.8986	0.9719

Table 1: *Classifier Statistics.*

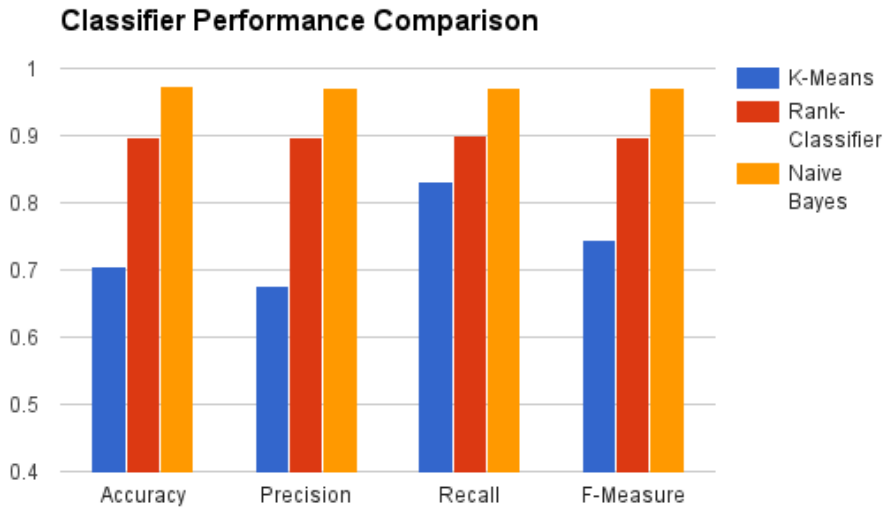

Figure 1: *Classifier Performance Comparison.*

As per the results, Naive Bayes outperforms the other two classifiers, and provides almost perfect results.(97% all measures.) The only few articles it missed were by marginal difference and overlapped topics. For example, 'Saudi ministry to employ women' is actually under business topic, but NB classified it as politics. The Ranking Classifier also perform well.(90% all measures) The tfidfie performs better than the normal tfidf(not explicitly shown). K-means algorithm has high recall and F-Measure. Since we used the combination of all three classifiers, the final result is mostly accurate.

For an example of recommendation, for the original article - "Wall Street cool to eBay's profit", a few recommendations were 1. GM issues 2005 profits warning, 2. European losses hit GM's profits and 3. Weak end-of-year sales hit Next. - which fall into same topics and close to the original article.

V. Conclusion

The recommendation system performs well, mainly dominated by Naive Bayes classifier. It can be further improved by assigning different weights to different part of speech after tagging is done. Also, instead of just relying on one choice, history of few previous choices can be used to recommend next articles belonging to multiple topics.

References

[Multinomial Naive Bayes: A Worked Example | Coursera] `https://class.coursera.org/nlp/lecture/28`

[Machine Learning Methods for Medical Text Categorization] http://ieeexplore.ieee.org.ezproxy.gl.iit.edu/stamp/stamp.jsp?tp=&arnumber=5232395&tag=1

[BBC Dataset] http://mlg.ucd.ie/datasets/bbc.html