Grenze International Journal of Engineering and Technology, Jan Issue

The Speculative Study on Machine Learning Algorithm for an Efficient Classification of Text Document

Mr. T. Murali Krishna[1] and Dr. T. Nalini[2]
[1]Research Scholar, Dept. of CSE, Bharath Institute of Higher Education and Research (BIHER), Chennai
[2]Professor, Dept. of CSE, Dr MGR Educational Research and Institute, Chennai

***Abstract*—Document Classification is a problem in information science. The text documents are assigned to one or more classes or categories for classification. Classifying text Documents using various Naive Bayes, Random Forest, Linear and Non-Linear SVM and compared these algorithms on different metrics like accuracy, F1-Score, training and testing time. Based on the analysis, SVM demonstrated that all of the other models when it comes to accuracy. Random Forest accuracy score was also quite good but took considerable time during training phase. From this we can gather the following information. There are three unique or distinct values in the category column which translates to the fact that there are three classes. "Administrative" category seems to be the most frequent category in this column also the count value confirms that there are no missing values in the dataframe now. It can be gathered clearly from the metrics that Linear SVM is clearly the winner in terms of strong accuracy score, training time and testing time. Random Forest and Non linear SVM classifiers' accuracy score is also quite appealing but the high magnitude of their training time is somewhat degrading. It has also been found and demonstrated that if we use lesser number of features (e.g. 5000), selected by applying chi-square test, then the accuracy scores remains the same. The further analyze the training and prediction time of the machine learning algorithms; it will come to further know that it would be greatly improved as well because of much less number of features. The results have been again validated by making use of cross validation. It demonstrated that the accuracy scores that got were equal to the ones got by applying cross validation.**

***Index Terms*— Random Forest Classifier, Multinomial Naive Bayes, Naive Bayes, Stochastic Gradient Descent.**

I. INTRODUCTION

The basic reason for data and information fusion is to reduce the uncertainty in our understanding and prediction of the state of certain aspect of the world. The text document classification system contains four different levels of scope that can be applied. The Document level is the first level where the algorithms are used to obtain the relevance for categorizing the document. Where the second level is the paragraph level, where the algorithms are used to acquires the relevance in single paragraph are categorized and therefore is also said that a portion of a document categorization. The third level is the sentence level, which are used to find the relevance in single sentence and categorizing the portion of the paragraph. The fourth level is the Sub-sentence level, where the algorithms are used to find the relevance in sub-expressions in a sentence.

Grenze ID: 01.GIJET.7.1.503_2

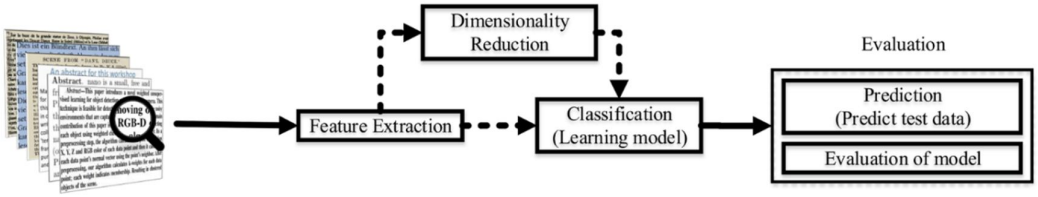

Figure 1. Text Document Classification System

A. Finding out the Value Distribution profile of the Category Variable

Its crucial to know in advance the basic statistics of the target/label variable so as to ensure that there is no issue of class imbalance there. For that, we need to calculate the count of each of the class label instances in our dataframe.

B. Lexical Analysis of the Text Data

As text data's pre-processing transforms the words/terms into respective columns (though in much improved approach e.g. only frequent terms and also the rare ones (TF-IDF) are transformed into columns, more on that later), so it's a good step to analyze the vocabulary richness of our data. If we just analyze the lexical richness of the text_description data, it will still give us much idea about how rich our data is in terms of unique vocabulary words.

II. Materials and Methods

A. Boosting

Ensemble learning for reducing variance in supervised learning. It belongs to the family of machine learning algorithms which convert weak learners to strong ones. As boosting is a strong learner it can able to classify the arbitrarily well-correlated and with true classification.

B. Naive Bayes Classifier

Naïve Bayes technique used for text and document categorization, where Naive Bayes Classifier (NBC) is a generative model that can widely use for Information Retrieval. which developed by using term-frequency (Bag of Word) feature extraction technique by counting number of words in documents.

C. Support Vector Machine (SVM)

The SVM was designed to find solution for binary classification issues, it have been used in multi-class problem based on authoritative technique.

D. Decision Tree

Decision tree algorithm for successful classification of text document are structured in the way of decomposition of data in hierarchical. The main issues is to determine the attributes of data points for tree creation which leads parent level and that will be in child level.

E. Dataset

The data-set had the following basic properties: It was provided in .csv format. The data-set simulated the real life scenario of jobs posted on a job portal and comprised of Job's title, Job's description along with its category As the data was labelled so in the context of machine learning, it was a Supervised Machine learning problem i.e. I had access to the data that was already correctly labelled and I had to train a model using this historical data. The main goal was to build a model that could accurately classify new and unseen data when it was input to it i.e. to assign proper class/label/category to a job posting when its input to the model. As the nature of the data was "text" so this project also involved extensive usage of text mining techniques as well.

Text in its basic form is unstructured and to develop predictive models, the data needs to be thoroughly pre-processed. So, the pipeline of developing models that: Data Profiling -> Data Cleansing -> Exploratory Analysis -> Data Pre-processing -> Feature Extraction and Selection -> Model Development -> Model Evaluation

When text data is pre-processed, the issue of curse of dimensionality usually appears i.e. data becomes highly multi-dimensional with lots of features ranging in thousands. Not all of those features are helpful and also it adversely affects the performance of classifiers as well so following the best practices, we opted for best-in-class feature extraction methods and also applied feature selection techniques so as to compile only those features that

will contribute in this prediction problem. For model development, we used and compared the following set of machine learning algorithms:

- Bernoulli Naive Bayes
- Multinomial Naive Bayes
- Random Forests

F. Naive Bayes

Naive Bayes is one of the most widely used classification algorithm in text mining applications. Based on Bayes theorem, this model makes the assumption that all the features are independent of each other and uses the probabilities of each attribute belonging to each class to make a prediction. The condition of independence may not be valid in many circumstances but as a base line model, it's a good starting point to test its performance on the provided data. There are two forms of Naive Bayes: Bernoulli (designed for Boolean /binary features i.e. just considers the presence or absence of a feature) Multinomial (which also considers the occurrence counts of the feature)

G. Linear SVM

SVM with non-linear kernel and compared these algorithms on different metrics like accuracy, F1-Score, training and testing time. As per my analysis, SVM outshines all of the other models when it comes to accuracy. Random Forests accuracy score was also quite good but took considerable time during training phase. For implementation, I used Python. Specifically, we used the following libraries/modules of Python for different set of tasks: Pandas, NumPy, SkLearn, Nltk, Matplotlib.

III. Experiment and Results

A. Experimental Setup

Initially the data has been loaded properly in the dataframe. The data-set had header rows in it as well which has been loaded in the dataframe as well. Though if required, we can assign different column names to this dataframe as well. the data-set has 5838 rows and 4 columns. So it can be gathered from the above snippet of code that the text_description column has 1 missing value. The next logical step should be to find out where is that missing value located. Thus it verifies the fact that there is a missing value in this column represent by NaN. Now there can be three approaches: rom this we can gather the following information:
There are three unique or distinct values in the category column which translates to the fact that there are three classes. "Administrative" category seems to be the most frequent category in this column also the count value confirms that there are no missing values in the dataframe now. Either we remove the row which has missing value or it will "imputate" the missing value or a simpler approach: Knowing that the "text_description" column contains text (str) so we can reload the data-set with this knowledge and clearly specifying that in str based missing values. Any of the above approach can be done and experimented with all of these but for this report, we followed the third method for the sake of brevity. Thus reloading the dataset with added parameters.

B. Data Pre-Processing

The data in its raw form isn't always suitable for developing analytical models. To make the raw data compliant for analysis, pre-processing steps have to be performed. The pre-processing steps depend largely on the type of analysis that one intends to perform for instance regression, classification or clustering. In our particular case, we want to perform multi-class classification and the data at hand is text. So to perform pre-processing, the following steps are performed: Converting all of the data into lower case. Stemming the words so as to further reduce the feature size. Also, in our given data set, we made use of both the given features i.e., title and job description to create new features and for pre-processing. I've found that if we use both titles and job description, better features are formed and overall accuracy score of classifier.

C. Steps involved in Automatic Document Classification

Step 1: Data Extraction

Initially the data are generated locally to train the proposed model. The function namely local-data.py are specified the dates and the key. This code will create the specified dataset (headers: id, text, label) and fix the category and labels file. The code will automatically increment the dates and keep on collecting the data specified by the iterations. There are multiple categories such as News, Sport, Television & radio, Environment, Science and design, Global, Food, Culture, Community, Money and Technology.

Step 2: Create a model using the extracted data
Later the dataset is uploaded and created the pipeline model using the pipeline.py. This file will save the pipeline in a folder and will show the accuracy of the model by using dataset for 80% training and 20% testing.
Step 3: Run stream_producer.py to use the extracted model to classify streaming data
Now the stream_producer.py to create for stream data and the spark_stream.py file to read the stream and load the created model and generate the prediction on the fly. Please, add the saved model full path in the spark_stream.py file.

D. Result Analysis

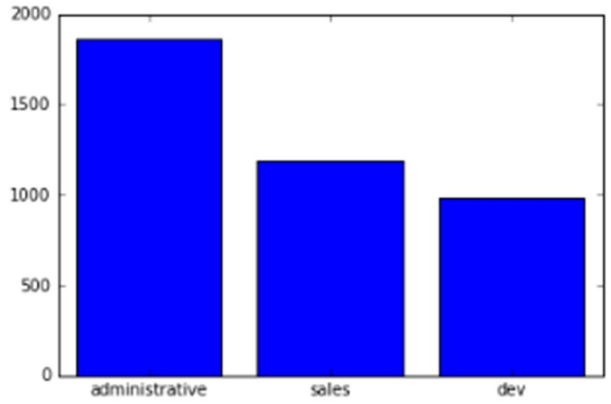

Figure 2. Distribution of Three Classes

The above plot that how the count of the three classes in the category column are distributed. As there is not much of a difference in the relative count of each of these classes so we can safely proceed to the model development. if there was a lot of imbalance, it also have to resort to under or oversampling during the training and testing phase so as to minimize the bias.
A very naive and base approach would be to use all of these 33k features in the classification. But it not recommended because of many classification algorithms can't work in such immensely high dimension data. Relatively better approach would be to use N top frequent words by following bag of words approach. But this tends to favour of those terms which are repeated in all of the corpus; which don't help in the overall classification objective. On comparison, a much better approach is to use TF-IDF. As we will see that number of unique vocabulary terms will be helpful in making comparison when we derive much less yet important features using TF-IDF. Applying Multinomial Naive Bayes: Bernoulli Naive Bayes just uses the fact that whether a feature is present or not. However if we somehow also take into account the occurrence weight or count of the feature as well (in our case, the TF-IDF weight of each feature), we can hypothesize that the performance of such classifier will be equally good, if not better. For this purpose, I've made use of the Multi-nominal Naive Bayes as Accuracy score 93 % in training time 33 sec and prediction time of 0.09 sec, whereas when applying Random Forest Classifier the ensemble learning methods for classification and regression tasks. In Random Forest, a subset of the training data is fit on a number of decision trees. Random Forests have the characteristic to minimize variance if its there in the data-set. There training time is generally quite higher which is one of their drawbacks to be used in production environment. However, the operations can be parallelized to reduce the training time. Also, greater the number of trees in the random forests, better will be the result however it comes with a trade off: the training time tends to increase as well. Also, if the number of trees tend to increase, the rate of improvement of accuracy may also tend to slow as well. I've used random forests with 50 trees. However, we can implement a routine to check the accuracy of the random forests by using different number of trees a 50 have training Time as 6.059000s where Accuracy Score as 0.9842.
The accuracy of this classifier is quite good but on the other hand, the training time and prediction time is quite higher compared to other classifier.
Applying cross validation is a model evaluation technique in which input training data is split into k sets/folds and model is learnt on k-1 sets and validated on the remaining set. It minimizes the risk of over-fitting and its main goal is to estimate how accurately a predictive model will perform in practice on un-seen data.

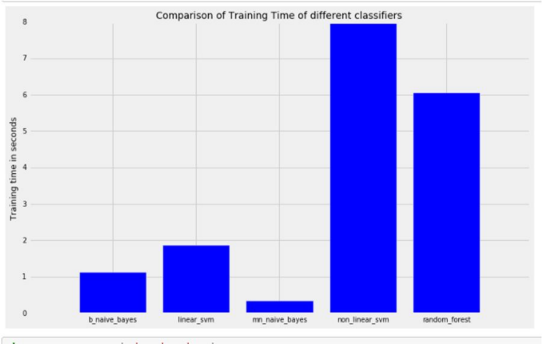

Figure 3. Comparison of Training Time of different classifiers

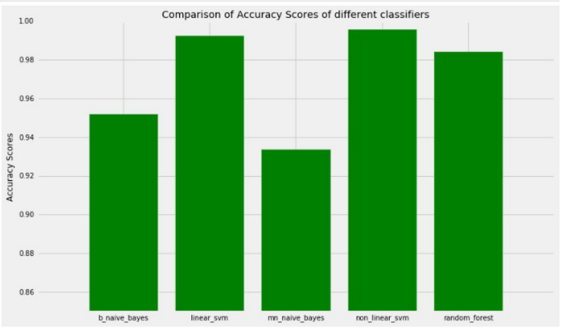

Figure 4. Comparison of Accuracy Scores of different classifiers

IV. Conclusion

It is clearly analysed from the metrics that Linear SVM is clearly the winner in terms of strong accuracy score, training time and testing time. Random Forest and Non linear SVM classifiers' accuracy score is also quite appealing but the high magnitude of their training time is somewhat degrading. It has also been found and demonstrated that if we use lesser number of features (e.g. 5000), selected by applying chi-square test, then the accuracy scores remains the same. If further analyzing the training and prediction time of the machine learning algorithms, we will come to further know that it would be greatly improved as well because of much less number of features. The results have been again validated by making use of cross validation. The accuracy scores that got are equal to the ones that is got by applying cross validation.

References

[1] D.L. Sackett, W.M. Rosenberg, J.A. Gray, R.B. Haynes, W.S. Richardson Evidence based medicine: what it is and what it isn't BMJ (Clinical research ed)., 312 (7023) (1996), pp. 71-72.
[2] F. Kboubi, A.H. Chabi, M.B. Ahmed (Eds.), Table recognition evaluation and combination methods, Document Analysis and Recognition, 2005 Proceedings Eighth International Conference on, IEEE (2005).
[3] H. Chao, J. Fan Layout and content extraction for pdf documents, Document Analysis Systems VI, Springer (2004), pp. 213-224
[4] M.E. Schaafsma, The Cochrane Collaboration Treasurer's Report, 2012.
[5] A. Constantin, S. Pettifer, A. Voronkov (Eds.), PDFX fully-automated PDF-to-XML conversion of scientific literature, Proceedings of the 2013 ACM symposium on Document Engineering, ACM (2013)
[6] K.G. Shojania, M. Sampson, M.T. Ansari, J. Ji, S. Doucette, D. Moher How quickly do systematic reviews go out of date? A survival analysis Ann. Intern. Med., 147 (4) (2007), pp. 224-233
[7] P. Bragge, O. Clavisi, T. Turner, E. Tavender, A. Collie, R.L. Gruen The global evidence mapping initiative: scoping research in broad topic areas, BMC Med. Res. Methodol., 11 (1) (2011), p. 92
[8] J.P. Higgins, S. Green, Cochrane Handbook for Systematic Reviews of Interventions Wiley Online Library (2008)
[9] D.D. Bui, S. Jonnalagadda, G. Del Fiol, Automatically finding relevant citations for clinical guideline development, J. Biomed. Inform. (2015).
[10] R.L. Summerscales, Automatic Summarization of Clinical Abstracts for Evidence-based Medicine, Illinois Institute of Technology (2013)
[11] K.-C. Huang, I.J. Chiang, F. Xiao, C.-C. Liao, C.C.-H. Liu, J.-M. Wong, PICO element detection in medical text without metadata: are first sentences enough?J. Biomed. Inform., 46 (5) (2013), pp. 940-946
[12] F. Boudin, J.-Y. Nie, J.C. Bartlett, R. Grad, P. Pluye, M. Dawes, Combining classifiers for robust PICO element detection, BMC Med. Inform. Decis. Mak., 10 (2010), p. 29
[13] H. Zhu, Y. Ni, P. Cai, Z. Qiu, F. Cao, Automatic extracting of patient-related attributes: disease, age, gender and race, Stud. Health Technol. Inform., 180 (2012), pp. 589-593
[14] D.P.A. Corney, B.F. Buxton, W.B. Langdon, D.T. Jones, BioRAT: extracting biological information from full-length papers, Bioinformatics (Oxford, England), 20 (17) (2004), pp. 3206-3213
[15] Verspoor K, Mackinlay A, Cohn JD, Wall ME. Detection of protein catalytic sites in the biomedical literature. In: Pac Symp Biocomput., 2013, pp. 433–444.
[16] J. Hakenberg, R. Leaman, N.H. Vo, S. Jonnalagadda, R. Sullivan, C. Miller, et al., Efficient extraction of protein–protein interactions from full-text articles, IEEE/ACM Trans. Comput. Biol. Bioinform., 7 (3) (2010), pp. 481-494.
[17] W. Hsu, W. Speier, R.K. Taira, Automated extraction of reported statistical analyses: towards a logical representation of clinical trial literature, in: AMIA Annual Symposium Proceedings/AMIA Symposium AMIA Symposium, 2012, pp. 350–359.

[18] B. de Bruijn, S. Carini, S. Kiritchenko, J. Martin, I. Sim, Automated information extraction of key trial design elements from clinical trial publications, in: AMIA Annual Symposium Proceedings/AMIA Symposium AMIA Symposium, 2008, pp. 141–145.
[19] S. Kiritchenko, B. de Bruijn, S. Carini, J. Martin, I. Sim ExaCT: automatic extraction of clinical trial characteristics from journal publications, BMC Med. Inform. Decis. Mak., 10 (2010), p. 56
[20] R. Kern, K. Jack, M. Hristakeva, M. Granitzer, TeamBeam meta-data extraction from scientific literature, D-Lib Magazine, 18 (7) (2012).
[21] M. Granitzer, M. Hristakeva, R. Knight, K. Jack, R. Kern (Eds.), A comparison of layout based bibliographic metadata extraction techniques, Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, ACM (2012)
[22] H. Han, C.L. Giles, E. Manavoglu, H. Zha, Z. Zhang, E. Fox (Eds.), Automatic document metadata extraction using support vector machines, Digital Libraries 2003 Proceedings 2003 Joint Conference on;, IEEE (2003).
[23] M.T. Luong, T.D. Nguyen, M.Y. Kan, Logical structure recovery in scholarly articles with rich document features, Multimedia Storage Retrieval Innovat. Digital Lib. Syst. (2012), p. 270.