AUTONOMOUS TAGGING OF STACKOVERFLOW QUESTIONS USING STATISTICAL METHODS

[1]RISHABH BASSI, [2]ROHAN PIPLANI, [3]RAVIJIT SINGH RAMANA, [4]JASMEET SINGH, [5]HARPREET SINGH, [6]PRASHANT SINGH RANA

[1,2,3,4,5,6] Computer Science Department, Thapar University Patiala-147004, Punjab, India
E-mail: akalharpreet@gmail.com

Abstract - Tagging of information plays an crucial role in indexing information. StackOverflow is one of the web portals that is based on query answering mechanism. It has a lot of data organized on the basis of tags. The research is focused on proposing an autonomous tagging based system. It uses the concept of 'Document-Term Matrix to predict various tags associated with a problem. This is done by choosing every tag having probability above a threshold level. The paper helps in showcasing the application of the machine learning models. It also establishes the a statistical relationship between precision and number of questions per tag. This results in optimizing the parameters i.e number of questions per tag and number of tags.

Keywords- Autonomous tagging; RTextTools; SVM; Random-forest Stackoverflow; Document-Term-matrix; multilabel-classification.

I. INTRODUCTION

Information sharing platform have become very popular platform for question and answer sessions. Examples include Quora, StackOverflow, Reddit and OpenEDX. While the quantity of information available on these websites has increased many folds but there is no efficient way to classify data as such that is automated. Most such websites ask users to tag their queries which is not an intuitive way to ask questions. As users might not tag the problem properly which further leads to ambiguity in data. It would be useful to automate the process of tagging as the means to classify information in an efficient manner. A system that supports autonomous tagging can improve the user experience by clustering information into dis-crete common topics. The other benefit is that the user can be recommended queries related to his own problem which could help him find the answer in an efficient and effective manner. The paper outlines a method for question and answer platforms that automatically allocates tags for a given query. We pro-pose a Document-Term matrix [16] based classifica-tion method to autonomously allocate tags to queries posted on any forum. The classifier was implemented on StackOverflow questions and was tested on the same. The remaining paper follows the following struc-ture. Section 2 discusses recent works in this area. Section 3 discusses the proposed work. Section 4 discusses the results and the last section concludes the paper.

II. RECENT WORK

Text classification is the method to classify text into various classes or tags based on their topic. It is based on application of natural language processing and machine learning algorithms. In brief, it is a multi-label classification problem. Work that has been done based on the survey of literature is the motivation behind this paper.

Algorithmic Programming Language Identification based approach is one such popular technique. Various syntax highlighting tools such as Google Code Prettify will automatically highlight syntax given some code. However, these tools do not actually identify languages; instead, they use heuristics that will make the highlighting work well. In the case of Google Code Prettify, broad grammars (such as C-like, Bash-like, and Xml-like) are preprogrammed. These grammars are then used to scan code, and the best matching grammar is used in highlighting. Clearly, languages that share a grammar cannot be distin-guished between. More relevant is SourceClassifier, which attempts to identify a programming language given some code. However, it relies on a simple Bayesian classifier. Its strength is therefore limited to the quality of training data, and it can easily be thrown off by strings and comments.

The other popular approach is automatic content tagging. In one paper [7], Xia et al. propose an automatic tag recommendation algorithm TagCombine. There are three components of TagCombine, each of which tries to assign the best tags to untagged objects: (1) multi-label ranking component, which predicts tags using a multi-label learning algorithm,
(2) similarity based ranking component, which uses similar objects to recommend tags, and (3) tag-term based ranking component, which analyzes the historical affinity of tags to certain words in order to suggest tags. The recommendation algorithm methodically computes various weighted sums of the three components to attempt to find the best overall model. Another algorithm by Wang et al. In second paper [21] proposes a tag recommendation system dubbed En-TagRec. The proposed EnTagRec computes tag

prob-ability scores using two separate methods, Bayesian Inference and Frequentist Inference, and then takes a weighted sum of the probability scores. Bayesian Inference relies on a posts textual data to compute the probability that a given tag is associated with the post. EnTagRec formulates posts into a bag-of-words model and then trains a Labeled Latent Dirichlet Allocation model which is used to compute tag probability scores for a post. The Frequentist Inference approach infers a set of tags after some preprocessing of a post, and then utilizes a network of tags to select additional tags that are similar to the ones in the set. The network of tags is constructed with the tags as nodes and weighted edges between two tags based on the Jaccard similarity of the posts that contain those tags.

III. EXPERIMENTAL SETUP

To conduct the research R library 'RTextTools [5] is required which is a supervised learning package for text classification. All the source code for the research is available on Git-hub repository [2]. The classifier was trained on stackoverflow dataset from Kaggle website [1]. There are around 37,000 unique tags in the dataset with around 37,00,000 questions from stackoverflow website [3]. The data is divided into 2 files which is questions and tags which are all related by a unique Id number assigned to each owner. Each question has an Id, OwnerUserId, CreationDate, ClosedDate, Score, Title, Body. For each entry in tags file there are two columns Id and Tag. The tradeoff between memory and increase in amout of data is becoming a bottleneck in computing. So, data was reduced to a single file which had id, title and tag as its constituents to make it easier to train the classifier. The glimpse of the final dataset is shown in the Table 1. The parameter chosen for evaluating the perfor-mance of the multi-label classification systems perfor-mance are precision, recall and F1 score [26]. Precision is the fraction of relevant instances among the re-trieved instances. Recall (the fraction of correct results returned) and F1 is defined as the harmonic mean of precision (the fraction of returned results that are correct). F1 score is one of the most common method to evaluate classification models performance. F1 is best at value 1 and worst at value 0. 2

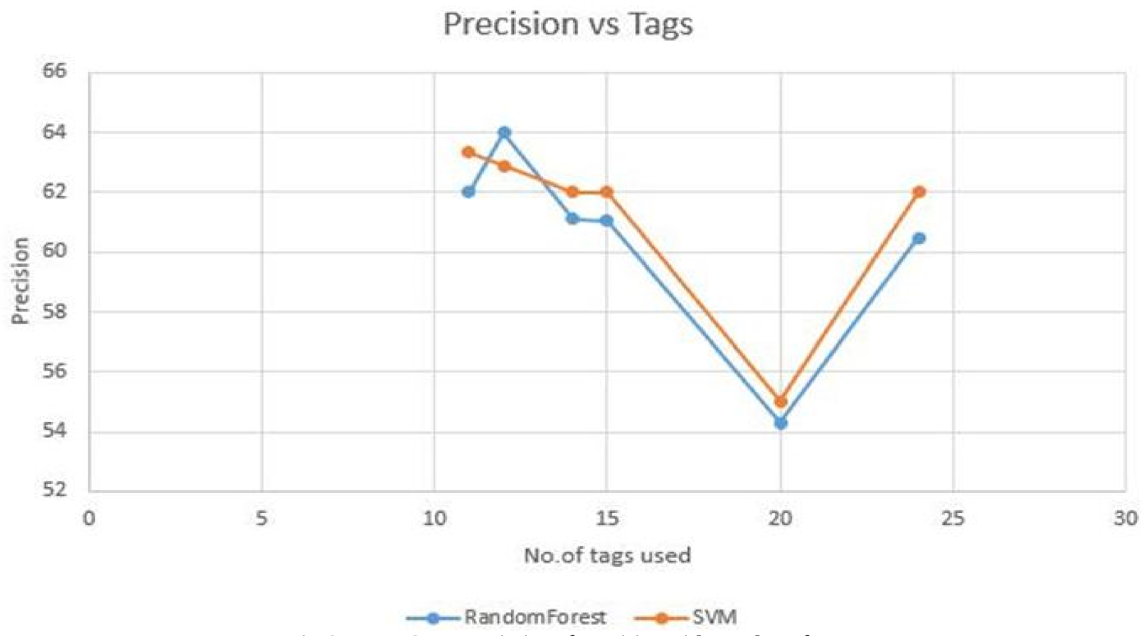

Fig. 2: RF vs SVM-Variation of precision with number of tags

IV. METHODOLOGY

Figure 1 explains the overall workflow of the research. The general workflow to train a classifier starts with loading the data into the workspace. In this paper StackOverflow dataset was used to train and test the classifiers. The data needs to be preprocessed due to the unbalancing of data. Unbalancing of data refers to the state wherein questions related to one tag are greater as compared to other tag. For example in case of StackOverflow data, javascript had 2,00,000 questions while tag of windows had only 10,000 questions [8]. So to balance the dataset number of questions were made consistent across each tag i.e. 10,000 questions per tag. After Preprocessing the data, a Document-term Ma-trix is generated. A Document-term matrix is the most common way of representing texts for proper computation. The matrix elements correspond to fre-quencies of occurence of word in a given file. It can be imported from a text dataset and uses a bag-of-words mechanism which means that it is independent of the order of tokens. This approach results in a matrix with document IDs as rows and vocabulary elements as columns. The creation of matrix also resulted in removal of stop words (refers to the most common words in a language that dont have a meaning by themselves) and used stemmer to return stem (a stem is a form to which affixes can be attached to the root word) words. All the sparse entries that is words with frequency less than 2 percent were removed [9]. After that, Document-Term Matrix was partitioned into a container using the method create container, which was basically a sequence of objects that was fed into the machine learning algorithms. The Document-Term matrix was partitioned by 70 percent data as training and the rest 30 percent was kept for testing. The classifiers used to implement proposed work are Random forest and SVM.

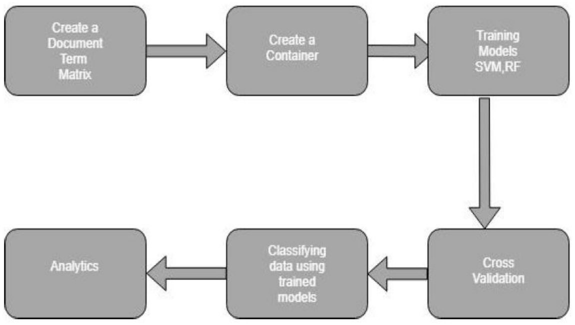
Create a Document Term Matrix
Create a Container
Training Models SVM,RF
Cross Validation
Classifying data using trained models
Analytics

Fig. 1: Flowchart for training multi-label classifiers

TABLE 1: Impact of attributes on model's performance

Id	Title	Body	Tag	Tagsid
80	SQLStatement.execute() multiple queries in one statement	I' ve written a database generation..	flex	11212
80	SQLStatement.execute() multiple queries in one statement	I' ve written a database generation..	actionscript-3	262
80	SQLStatement.execute() multiple queries in one statement	I' ve written a database generation..	air	699
90	Goodbranching and merging tutorials for TortoiseSVN	Are their any good tutorials..	svn	31432
90	Goodbranching and merging tutorials for TortoiseSVN	Are their any good tutorials..	tortoisesvn	32802
90	Goodbranching and merging tutorials for TortoiseSVN	Are their any good tutorials..	branch	3952
90	Goodbranching and merging tutorials for TortoiseSVN	Are their any good tutorials..	branching-and-merging	3954

TABLE 2: Impact of attributes on model's performance

No.of Tags used		Random Forest			SVM	
	Precsion	Recall	F1-Score	Precsion	Recall	F1-Score
11	62	58.6	59.2	63.33	59.86	60.2
12	64	60.1	62.7	62.9	58.6	60.1
14	61.12	57	58.2	62	58.7	59
15	61.04	57.1	58	62	58.5	58.7
20	54.3	50.4	49.6	55	50	51.2
24	60.47	57.9	55.1	62	58.6	57.6

V. PREDICTION MODELS

Classifiers such as Random-Forest, SVM, Neural-Net, SLDA, MAXENT etc. were trained on stackoverflow dataset. The results showed that Random Forest and SVM were the most accurate out of all classifiers.

1. Random Forest: It works on the concept of Bagging method which is a combination of learning models. Random Forest is a supervised learning algorithm which makes use of multiple Decision Trees[23] . It groups the result of various decision trees to get a more accurate and stable prediction. We can futher increase performance of our model by tuning different parameters such as n estimators, random state, max features etc. One advantage of random forest is that it can be used for both classification and regression problems. Random Forest also solves one of major problems of machine learning i.e. overfitting. As there are enough trees in the forest, the classifier wont overfit the model [15]. It contains input vector X = x_1; x_2; :::; x_p, where a p-dimensional input vector that works in build-ing a forest. Inside the forest a set of K trees T1(x); T2(x); :::; Tk(x), the output of each tree estimates the actual value aY^ 1 = ^ T 1(X); :::; Y m = Tm(X), where m = 1; :::; K. The final result of it is the mean of all the values predicted by different trees. $T_m(X)$, where m = 1; :::; K. The final result of it is the mean of all the values predicted by different trees.

$$\text{Estimate}_{RF}(X) = \frac{1}{k} \sum_{=1}^{kX} \hat{y}_k(X) \qquad (1)$$

The training dataset is independently taken from the input and output

D= D_1; D_2; :::; D_n = (x_1; y_1); (x_2; y_2); :::; (x_n; y_n), where x_i; i = 1; :::; n, is the training dataset for input vector and y_i; i = 1; :::; n; is training dataset for output vector. Each tree is grown using the training approach. The estimated error and accuracy is evaluated for the random forest using the minimization of mean square error (M SE). Determining the optimum trees in forest focuses on M SE. Testing data is communicated using each split node, by sending it either to right or to left child until ending up at a leaf node.

$$\underset{\wedge}{M\,SE} \quad M\,SE^{oob} = \frac{1}{n} \sum_{=1}^{n} (Y(X_i)\ Y_i)^2 \qquad (2)$$

Here, ^ represents estimated output of trees Y (X_i) in forest corresponding to a given input sample, Y_i shows observed output and n shows a total number of samples. However, a random forest has difficulties in select-ing the number of trees. In this work, we have taken a number of trees =100[4].

2. SVM: Another classifier used was Support Vector Machine which also is a supervised learning model It also analyzes data used for both classification and regression analysis. Our training data consists of features which belong to one or other of two cate-gories. An SVM training algorithm builds a model out of training data and this model is used to classify test data to one or the other category. Performance of SVM depends on selection of kernel, kernels parameter and margin parameter C. It also has the advantage of avoiding overfitting. And as SVM models support kernels, we can even model non-linear relations. It is also more robust as it maximizes margin [22]. Support Vector Machines can be thought of as linear classifier, in a way that produces classifiers with the-oretical guarantees of good predictive performance. A set of points (x_i; y_i), i = 1, l, where y_i 2 (-1, +1) are class labels, is called linearly separable if a linear classifier can be found so that y_i*f(x_i) > 0, i = 1, . . . , l. A hyperplane < w ; x > +b = 0; jjw jj = 1 is called -margin separating hyperplane if y_i(< w ; x > +b) >= for all (x_i; y_i) in set S. Here (clearly > 0) is the margin [17]. Thus we would like to find a classifier with the largest margin that still correctly separates the train-ing points. After training of classifiers they were tested using test dataset ~~u~~sing inbuilt function of classify model in RTextTools. The most crucial step during the machine learning process is interpreting the results, and RText-Tools ~~p~~rovides a function called create analytics() to help users understand the classification of their test set data. The summary function is used to on the out-put of create analytics to generate a comprehensive result.

VI. RESULTS ANALYSIS AND VALIDATION

The evaluation metric to measure the performance of our multi-label classifier is Precision, Recall and F1 score [25]. The research work had 10-fold cross validation to ensure a consistent level of performance was achieved. It also helped to iron out the errors

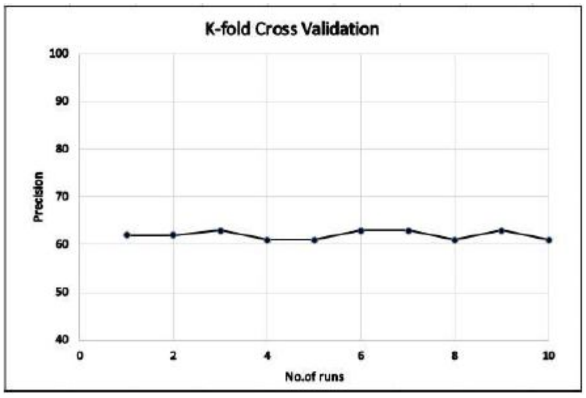

Fig. 3: K-Fold Cross Validation

[19]. The results reported for the predictions that were made by varying the number of tags with aforemen-tioned classifiers. Using the unbalanced and biased dataset, the precision was 2.7 percent. After balancing of the dataset, which was done by creating a dataset in which each tag has 10,000 questions corresponding to it the number of tags were varied. Since 70 percent dataset was used for training the data, the number of questions to be trained on was 7,000 and classifiers were tested on the remaining 3,000 questions. With 11 tags, the precision of Random forest was 62 percent while for SVM was 63.33 percent. By increasing the number of tags to 12, the precision for SVM increased to 64 percent while for random forest increased to 62.9 percent. By increasing the number of tags to 14, the precision of SVM model dropped to 62 percent and Random forest to 61.12 percent. Increasing the number of tags further to 15 had no effect on the precision for SVM and Random forest as the results remained on a similar level i.e. 62 percent and 61.04 percent. Increasing the number of tags to 20 led to further decrease in precision i.e. for SVM the precision was 55 percent and for Random Forest it was 54.3 percent. Going even further with number of tags i.e. 24 tags, the precision of Random forest was 60.47 percent and of SVM was 62 percent.

Table 1 and Fig 2 summarize the entire research results.

Data is divided into two parts i.e training and testing. Training and testing both should contain maximum data points. So, if points are removed from training to testing, there is loss of data points in training which can lead to underfitting. Now, to overcome this problem, k-fold cross validation is used [12]. The basic idea is that dataset is partitioned into k-bins of equal size. Here, it is divided into 10 bins where each bin contained 10% of the dataset. Each time 3 random bins is selected as the testing set and remaining 7 bins as the training dataset. This process is repeated 10 times which resulted in different accuracies of the different prediction models. Fig 3 shows the k-fold cross validation of the precision. From the graph, it is clearly seen that accuracy is not varying to a great extent from which it can be concluded that model is robust.

CONCLUSION AND FUTURE WORK

The work done is based on Document Term Matrix based classification systems. The proposed classifica-tion system which gave the best result was imple-mented using a random forest and svm model and carefully varied tag and questions per tag to achieve an optimal precsion of 64%, recall of 60.1% and F1 score of 62.7% for a sampled portion of the dataset with 12 most popular tags and 10,000 questions per tag. As future work, we plan to increase the precsion by taking similarity between questions which can be measured using jaccard distance [18] and add this value to the term document matrix [6]. It will group similar questions and thereby increase the tagging precsion. Morever we used some of the popular tags because covering greater number of tags decrease the accuracy. So our next plan would be primilary to cover more popular tags thereby increasing the scope of our prediction system. We also plan on exploring some advanced techniques such as deep learning to get a better grasp of dataset [14]. Since dataset is quite rich in knowledge deep learning methods will surely extract more information than existing statistical methods [13].

REFERENCES

[1] www.kaggle.com/stackoverflow/stacksample. 2016.

[2] github.com/bassirishabh/stackoverflow. 2018.

[3] www.stackoverflow.com. 2018.

[4] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[5] Loren Collingwood, Timothy Jurka, Amber E Boydstun, Emil-iano Grossman, WH van Atteveldt, et al. Rtexttools: A supervised learning package for text classification. 2013.

[6] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. pages 126–135, 2006.

[7] M Eric, A Klimovic, and V Zhong. # ml# nlp: Autonomous tagging of stack overflow questions, 2014.

[8] Amir Fallahi and Shahram Jafari. An expert system for detection of breast cancer using data preprocessing and bayesian network. International Journal of Advanced Science and Technol-ogy, 34:65–70, 2011.

[9] Ingo Feinerer. Introduction to the tm package text mining in r. 2013-12-01]. http://www, dainf, ct. utfpr, edu. br/-kaestner/Min-eracao/RDataMining/tm, pdf, 2017.

[10] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. pages 137–142, 1998.

[11] Thorsten Joachims. Transductive inference for text classification using support vector machines. 99:200–209, 1999.

[12] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. 14(2):1137–1145, 1995. 5

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Ima-genet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436, 2015.

[15] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. R news, 2(3):18–22, 2002.

[16] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. International Journal on Artificial Intelligence Tools, 13(01):157–169, 2004.

[17] S. Nashat, A. Abdullah, S. Aramvith, and M. Z. Abdullah. Original paper: Support vector machine approach to real-time inspection of biscuits on moving conveyor belt. Comput. Electron. Agric., 75(1):147–158, January 2011.

[18] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. Using of jaccard coeffi-cient for keywords similarity. In Proceedings of the International MultiConference of Engineers and Computer Scientists, volume 1, 2013.

[19] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. pages 532–538, 2009.

[20] Sebastian Schuster, Wanying Zhu, and Yiying Cheng. Predict-ing tags for stackoverflow questions. CS229 Projects, Stanford university, 2013.

[21] Logan Short, Christopher Wong, and David Zeng. Tag recom-mendations in stackoverflow. Technical report, San Francisco: Stanford University, CS, 2014.

[22] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. Neural processing letters, 9(3):293–300, 1999.

[23] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Ran-dom forest: a classification and regression tool for compound classification and qsar modeling. Journal of chemical information and computer sciences, 43(6):1947–1958, 2003.

[24] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. International Journal of Data Warehousing and Mining, 3(3), 2006.

[25] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. International Journal of Data Warehousing and Mining, 3(3), 2006.

[26] Yiming Yang and Xin Liu. A re-examination of text categorization methods. pages 42–49, 1999.

★★★