Nat. Volatiles & Essent. Oils, 2021; 8(6): 1281-1288

NVEO
Natural Volatiles &
Essential Oils

Multi-Label Classification System That Automatically Tags Users' Questions To Enhance User Experience

Aluri Anuradha[1], Dr. Kanakam Siva Rama Prasad[2], Srinivasa Rao Madala[3]

[1] M.Tech., Scholar, Department of Computer Science and Engineering,

[2] Associate Professor, Department of Computer Science and Engineering,

[3] Professor & HOD, Department of Computer Science and Engineering

PACE Institute of Technology and Sciences

Abstract:

Information tagging is critical to the indexing process. Web portals based on question response mechanisms include Stack Overflow. There's a lot of information here, and it's arranged using tags. The study's goal is to develop a system that relies on self-tagging to track objects. It makes advantage of the 'Document-Term Matrix' idea to foretell different tags connected to a problem. This is accomplished by selecting all tags with probabilities greater than a predetermined threshold. The research contributes to demonstrating how machine learning models may be put to use. By doing so, it also creates a statistical link between accuracy and the amount of inquiries per tag. There is a benefit in optimising the parameters, such as how many tags there are for each question.

1. Introduction:

To begin, platforms for exchanging information and doing Q&A sessions are becoming increasingly popular. Quora, StackOverflow, Reddit, and OpenEDX are a few of the many examples. Although the amount of data accessible on these websites has multiplied several times, there is no effective solution to categorise data in this way that is automated. Users are typically required to tag their searches, which is counterintuitive. As a result of people failing to correctly categorise the issue, data becomes even more ambiguous. To efficiently classify information, automating the tagging process would be beneficial. By grouping information into discrete common categories, a system that allows autonomous tagging can enhance the user's experience. In addition, the user can be offered inquiries relating to his own problem, which will help him locate the solution quickly and effectively.- For question and answer systems, the article proposes a tag allocation approach that works automatically. The rise of online education has led to an increase in the use of question-and-answer forums as a source of information. You may find them on sites like Stack Overflow, Quora, as well as MOOCs from providers like Coursera and OpenEdX. There is an increasing amount of material accessible on these forums, but no efficient or

automatic technique of organising and classifying it so that it can be shown to users in an intelligible manner is available at this time. It would be helpful if a question submitted on a forum could automatically deduce and tag the topic. By grouping questions about common themes together for users to peruse, a system that automatically infers the topic of a question can improve the user experience on online forums. Some forums, like Quora, require members to explicitly insert tags associated with their queries in order to facilitate the grouping of common posts. Manually marking a post, on the other hand, is a hassle for users and reduces the overall quality of the service. We want to build a platform where postings' tags may be automatically deduced. To that aim, we've developed a system that automatically identifies questions posted on a forum based on many labels. We use a dataset of Stack Overflow questions to train and test our classifier.

2. Literature Work

Using text categorization, you may divide up a text's subject matter into several classes or tags. It utilises machine learning and natural language processing to analyse text. It's a classification issue with several labels, to put it simply. This work is motivated by research done after a thorough review of the available literature.

One well-liked strategy is focused on the identification of algorithmic programming languages. Google Code Prettify and other syntax highlighting utilities automatically highlight syntax in provided code. Instead of using heuristics to highlight words, these systems don't truly detect languages. Broad grammars (such as Clike, Bashlike, and Xmllike) are preprogrammed in Google Code Prettify.

As a result of this, grammars are employed to scan code, with the best matching grammar being utilised for highlighting.

Languages that share a grammar are indistinguishable from one another, it is clear. Source Classifier, on the other hand, tries to identify a programming language based on a sample of source code. A basic Bayesian classifier is all that's required. As a result, its usefulness is constrained by the quality of the training data, and strings and comments can quickly derail it.

Automatic content tagging is an alternative strategy that's becoming increasingly popular. Tag Combine is a method proposed by Xia et al. in one publication [7].

Using Tag Combine, you may apply the best tags to untagged objects by combining its three components. Multi-label ranking component that predicts tags using a multi-label learning algorithm, similarity-based ranking component that recommends tags based on similar objects, and tag-term ranking component that analyses the historical affinity of tags to certain words in order to suggest tags are the three main components. To select the optimum overall model, the recommendation algorithm painstakingly computes alternative weighted sums of the three components.

Wang et al algorithm .'s En-TagRec, a tag recommendation system, is proposed in the second paper [21]. Using Bayesian Inference and Frequentist Inference, EnTagRec calculates tag prob-ability scores and then takes a weighted sum of the probability scores.

As the name implies, Bayesian Inference makes use of the linguistic data in a post to estimate the likelihood that a certain tag is attached to it. Labeled Latent Dirichlet Allocation is used to generate post tag probability scores using EnTagRec's bag-of-words formulation. This method uses a preprocessed post to derive a set of tags, and then it uses a network of tags to choose additional tags that have

similarities with the ones already found. When building the tag network, nodes are the tags themselves and edges are calculated based on the Jaccard similarity of articles that contain those tags.

3.Methodology

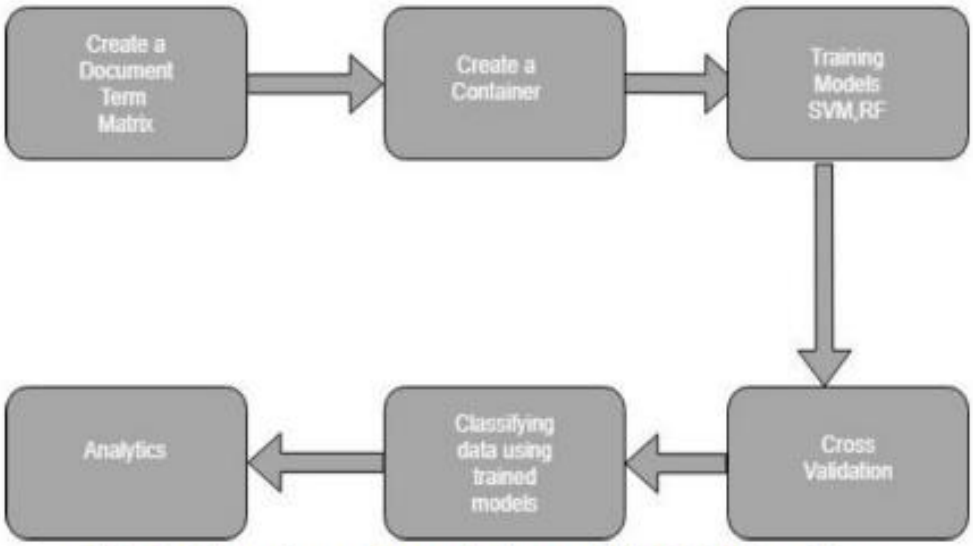

Fig. 1: Flowchart for training multi-label classifiers

Figure 1 depicts the research's overall procedure. Loading the data into the workspace is the first step in training a classifier. The classifiers in this research were trained and tested using the StackOverflow dataset. The unbalanced data need preprocessing. Unbalanced data refers to a situation in which inquiries pertaining to one tag are more numerous than those pertaining to another. For instance, javascript had 2,00,000 StackOverflow questions whereas the category of Windows only had 10,000 queries [8]. To maintain the dataset's balance, the number of questions was fixed at 10,000 for each tag. A Document-term Ma-trix is created once the data has been preprocessed. The most popular representation of texts for proper calculation is a Document-term matrix. In the following file, the matrix entries represent the frequency with which certain words occur. It employs a bag-of-words technique and may be imported from a text dataset, thus it doesn't care about token order.

Thus, a matrix with document IDs as rows and vocabulary components as columns is created using this strategy. Removed stop words (words that don't have a meaning on their own) and utilised stemmer to return stem words (a stem is an affix that may be appended to the root word) were the results of the development of matrix, as well. Deleted were all of the sparse entries, which were terms with a frequency of less than 2%. Using the method build container, Document-Term Matrix was then divided into a container that was supplied into the machine learning algorithms as a series of objects. Seventy percent of the Document Term matrix's data were used for training, while the remaining thirty percent were preserved for testing.

4. Prediction Models

The stackoverflow dataset was used to train classifiers including Random-Forest, SVM, NeuralNet, SLDA, and MAXENT, among others. Overall, Random Forest and SVM performed best, followed closely by Random Forest and Support Vector Machine (SVM).

Random Forests: A mixture of learning models is used in the Bagging approach. Using several Decision Trees, a supervised learning system called Random Forest[23] learns by doing. Various decision trees' results are combined to provide a more accurate and reliable forecast. By adjusting parameters like n estimators, random state, and max features, we can get even better results from our model. Random forest has the benefit of being applicable to both classification and regression tasks. Additionally, Random Forest eliminates overfitting, which is a common problem in machine learning. The classifier won't overfit the model since the forest has enough trees [15]. A p-dimensional input vector is used in the construction of a forest, and it contains the input vector X = x1; x2;::; xp. For each tree in the forest, the output estimates the real value aY 1 = T 1(X);:::; Y m = Tm(X), where m=1. In the forest, there are K trees T1(x), T2(x), and so on. The final result is the average of all the anticipated values from various trees. Tm(X), where m is a positive integer, and :::; K is the result. The final result is the average of all the anticipated values from various trees.

The input and output data are used to create the training dataset.

There are two training datasets, one for the input vector and the other for the output vector, with the difference being that the training dataset for the input vector is xi; I = 1 and the training dataset for the output vector is yi; I = 1 and the training dataset for the output vector is yn; yn. The training method is used to grow each tree. Minimization of mean square errors is used to assess the estimated error and accuracy for the random forest (M SE).

M SE is used to find the best forest trees. Testing data is sent to the right or left child of each split node until it reaches the leaf node.

If we take a given input sample and estimate the forest's tree Y (Xi) output using this equation (), we get the following results: Yi represents observed tree Y output, and n indicates the total number of samples taken. However, choosing the number of trees in a random forest is tricky. We've used a large number of trees in this study =100[4].

Support Vector Machine (SVM): Another classifier utilised was a supervised learning model called Support Vector Machine. Classification and regression analysis data are both analysed using this programme. The characteristics in our training data may be divided into two groups. This method creates a model from practise data and uses it to sort test data into one of two groups. The SVM's performance is determined by the kernel, kernel parameter, and margin parameter C that are all chosen. Overfitting may be avoided by using this method. We can even model nonlinear relationships using SVM models, which support kernels. As a result, it's more durable, thanks to its emphasis on margin [22]. Using SVMs, you may create linear classifiers that are theoretically guaranteed to perform well in the real world. If a linear classifier can be discovered such that yi*f(xi) > 0, I = 1,..., l, then a set of points (xi; yi), I = 1, is said to be linearly separable. As long as yi(-margin separating hyperplane) >= for all the (xi, yi) values in the set S, the hyperplane has the property of being dubbed the "minus margin" or "-margin" separating hyperplane. The margin is shown above as [17] (obviously greater than zero). A

classifier with the biggest margin that successfully differentiates training points is what we're looking for. After classifiers were trained, RTextTools' inbuilt classify model function tested them against a test dataset. When it comes to machine learning, the most critical stage is interpreting the findings, and RText-Tools' generate analytics() function makes it easy for users to do just that. In order to obtain a thorough result, the summary function is used to the analytics output.

5. Results

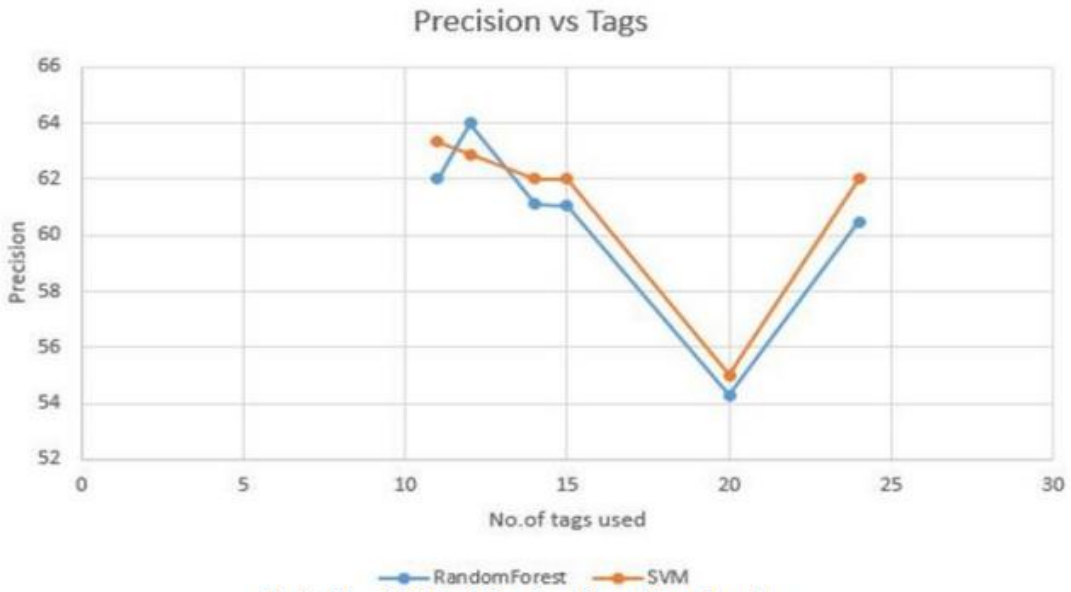

Fig. 2: RF vs SVM-Variation of precision with number of tags

Precision, Recall, and F1 score [25] are the assessment metrics we used to gauge the classifier's performance. To guarantee a constant level of performance, the research was 10-fold cross-validated. It aided in the correction of blunders as well.

K-Fold Cross Validation. These are the outcomes of predictions produced using different combinations of tags and the aforementioned classifiers. The accuracy was 2.7% while using the skewed dataset.

After the dataset was balanced, the number of tags was adjusted by producing a dataset with 10,000 questions matching to each tag. Since 70% of the dataset was used to train the data, there were 7,000 questions to train on and 3,000 questions to test the classifiers on. Random forest had a precision of 62 percent with 11 tags, whereas SVM had a precision of 63.33 percent. SVM precision improved by 64% while random forest precision grew by 62.9 percent when tags were raised to 12. As the number of tags was increased to 14, the SVM model's precision fell to 62% and the Random forest model's precision fell to 61.112%. In SVM and Random forest, increasing the number of tags to 15 had no influence on precision; the results stayed at the same level, at 62 percent and at 61.04 percent, respectively.

Increasing the number of tags to 20 resulted in a further loss of precision, resulting in a precision of 55% for SVM and 54.34% for Random Forest. SVM had a 62 percent precision and Random forest had a 60.47% precision while using 24 tags.

To summarise the findings, see Fig. 2 and Table 1.

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

There should be as many data points in the training as possible. Taking points from training to testing means losing data points in training, and this can lead to underfitting and overfitting. K-fold cross validation [12] is now being utilised to solve this issue. The primary concept is to divide the dataset into equal-sized k-bins. Using this method, the dataset was separated into 10 groups of equal size. 3 random bins are used for testing, while the remaining 7 bins are used to train the algorithm. This procedure was performed ten times, resulting in varying degrees of accuracy for the various prediction models. Figure 3 demonstrates the precision's k-fold cross validation. The graph clearly shows that accuracy does not fluctuate greatly, hence the model may be considered robust based on this data.

6. **Conclusion**

Document Term Matrix-based categorization techniques underlie the completed work. Random forest and svm models were used to implement the proposed classification system, which yielded the best results. Tags and questions per tag were varied to achieve an optimal prediction of 64%, recall of 60%, and an F1 score of 62.7 percent for a sampled portion of the dataset with the 12 most popular tags and 10,000 questions per tag. Future work will include increasing the precision by utilising jaccard distance [18] to quantify question similarity and adding that value to the word document matrix [21,22]. It will make tagging more accurate by putting related questions together. Furthermore, we made use of a few well-known tags in order to reduce the overall accuracy. As a result, our next step is to expand our prediction system's coverage to include more popular tags. Using modern approaches like deep learning, we hope to get a deeper understanding of the dataset [14-19]. Deep learning approaches will undoubtedly extract more information from this dataset than current statistical methods since the dataset is so rich in knowledge [23-26].

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. Crossvalidation. 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. Madala, S. R., Rajavarman, V. N., & Vivek, T. V. S. (2018). Analysis of Different Pattern Evaluation Procedures for Big Data Visualization in Data Analysis. In Data Engineering and Intelligent Computing (pp. 453-461). Springer, Singapore.
25. Madala, S. R., & Rajavarman, V. N. (2018). Efficient Outline Computation for Multi View Data Visualization on Big Data. International Journal of Pure and Applied Mathematics, 119(7), 745-755.
26. Vivek, T. V. S., Rajavarman, V. N., & Madala, S. R. (2020). Advanced graphical-based security approach to handle hard AI problems based on visual security. International Journal of Intelligent Enterprise, 7(1-3), 250-266.