Application of Machine Learning Algorithms in Automatic Question Tagging

Medha Joshi
Department of Artificial Intelligence & Data Sciences
IGDTUW,Delhi, India
medha027mtaids22@igdtuw.ac.in

Ritika Kumari
Department of Artificial Intelligence & Data Sciences
IGDTUW[1], USICT[2], GGSIPU[2]
Delhi[1,2], India
ritikakumari@igdtuw.ac.in

Poonam Bansal
Department of Artificial Intelligence & Data Sciences
IGDTUW,Delhi, India
poonambansal@igdtuw.ac.in

Amita Dev
IGDTUW, Delhi, India
vc@igdtuw.ac.in

***Abstract*— Stack Overflow, a widely used internet platform for asking coding doubts, learning coding and sharing knowledge, facilitates public engagement through its question-answering forum. Users on Stack Overflow are required to manually enter a minimum of one tag for the questions they post. Tagging involves associating keyword with the sentence or question, aiding in categorization and accessibility. Analysis of tags on the platform revealed that a significant number of questions are labeled with multiple tags, and inaccuracies in tagging are common. This situation hinders users in searching for suitable tags. The primary objective of this research is to investigate, analyze and compare methods for developing an auto-tagging system using machine learning and deep learning techniques, along with Natural Language Processing based data preprocessing steps. The outcomes of the study yielded satisfactory results with opportunities for improvement.**

Keywords—Prediction, Stackoverflow, Multilabel Binary classification, Natural Language Processing.

I. INTRODUCTION

As the virtual population and internet user base continue to grow, the volume of text data, particularly in the form of Q&A, has seen a substantial increase. The objective we aim to address involves the tagging or labeling of provided text, particularly questions. From the Stack Overflow platform. To enhance user navigation, these platforms frequently encourage users to label their posts with relevant tags, enabling individuals with expertise in specific domains to address questions suited to their knowledge. Additionally, it allows users interested in particular subjects to efficiently locate posts most relevant to their interests. While experienced users may find tagging a straightforward task, newcomers may encounter challenges in the process. The segregation and classification of questions become imperative to prevent questions from getting lost among a multitude of unanswered queries.

Question classification is crucial for efficient categorization, ensuring easy exposure to the intended target users and facilitating more frequent answers. There are three primary methods for tagging: (1) manual tagging, (2) semi-automatic tagging, and (3) automatic tagging. Manual tagging has certain drawbacks, such as the necessity for taggers to possess expertise in the subject when manually assigning knowledge units to questions. Additionally, variations in perspectives among different taggers can lead to subjective differences, posing challenges in maintaining consistency and accuracy. Semi-automatic tagging involves content analysis, generating tags that require further user processing, making human assistance essential. On the other hand, automatic tagging processes content without human involvement, yielding standardized and consistent outcomes at reduced costs.This system overcomes the limitations of manual and semi-automatic tagging by being faster and less susceptible to errors or biases.

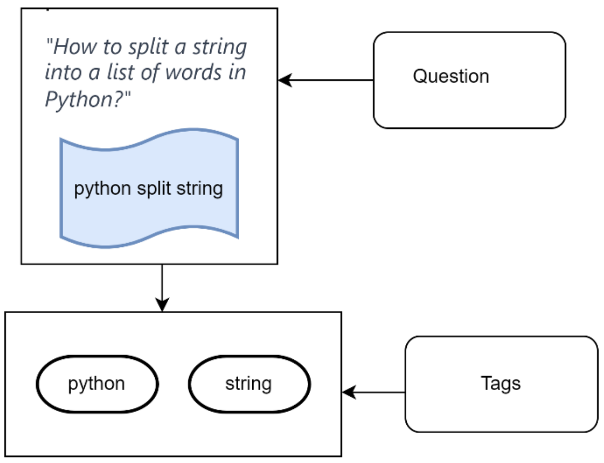

Fig 1 : Stack overflow Question with Tag

Due to the impracticality of using the entire dataset from the website, a small portion (10 percent of the platform's questions) is utilized, obtained from Kaggle. The paper outlines the following three key objectives:

- Implementing a robust and well-structured text preprocessing pipeline to optimize the efficiency of the classification task in the context of Natural Language Processing (NLP)
- Modify Binary classification-focused models to enhance their applicability and effectiveness in multilabel classification problem of Tagging.
- Evaluating the effectiveness of different machine learning tagging algorithms by assessing their performance across diverse evaluation parameters.

II. LITERATURE SURVEY

In the past decade, numerous studies have explored automated topic tagging. Earlier efforts, such as those by Robinson and Guo at Google Cloud [1] and another model [2] utilized a SVM machine learning centred model to establish a maximum margin model for question tagging. There is a similar approach that leverages K-nearest neighbours classifier and machine learning algorithm Random Forest[3] for tagging single tags, but it falls short by neglecting multi-label tags and lacks a comparative analysis with various other ML models, a gap addressed in our research. An alternative model, TagCombine [4], proposed three distinct but composite methods: multi-label ranking to consider tagging recommendation, similarity-based ranking to consider tags from similar objects and tag-term-based ranking. Additionally, EnTagRec [5], another noteworthy approach, employs Bayesian Inference techniques to compute score of probability for tags. It then determines a weighted sum of the obtained scores. One noteworthy unsupervised approach has proven effective in the context of tagging. Miotto and Weng [6] introduced a methodology for assigning an index to clinical text by extracting frequent tags from the text itself.

Furthermore, Advanced pre-trained natural language processing model BERT based techniques are used in [7], but are computationally expensive and resource-intensive, typically requiring powerful hardware such as GPUs or TPUs and substantial training time.

These diverse models and approaches contribute unique solutions to the question tagging problem. The existing body of research motivates us to seek a distinct and more effective algorithm. Consequently, the primary objective of this paper is to present an enhanced algorithm capable of efficiently tagging diverse question data.

.

III. MATERIALS AND METHODS

This model delineates the strategy employed to execute the current task. The following steps are employed our preprocessing pipeline.

A. *Dataset: Questions and Tags*

The mentioned Stack Overflow dataset constitutes a substantial and diverse collection of information. It serves as a repository of questions from the Stack Overflow website, a platform tailored for professional and learner programmers seeking answers. The dataset encompasses seven key columns: Question ID, Creation Date, Owner's ID, Closure Date, User-assigned Score, Question’s Title, and Question’s Body. Tags represent categorizations for every question, with the flexibility for each question to be associated with multiple topic tags therefore every question ID has not just one assigned tag but multiple tags within the Tags dataset. The dataset comprises around 1.2 million questions, each having at least one associated tag. Notably, questions are assigned to approximately 14000 unique categories of tags.

B. *Data Cleaning*

First of all steps to drop NaN values were carried out .Few columns such as Owner's ID, Closed date and Creation date are not pertinent to the prediction of tags and are therefore not required and dropped.

The Tags dataframe contains various tags in separate rows, each associated with a corresponding question ID. This Tags dataframe was devoid of any NaN values, simplifying the processing. To enhance the utility of the tags, they were categorized based on their assigned question IDs. Afterward, the categorized tags were consolidated with the Question dataframe using the question IDs as the linking reference. Question IDs were then transformed into lists, facilitating better comprehension and visualization of data. From EDA it was observed that the average number of tags for each question was 3.

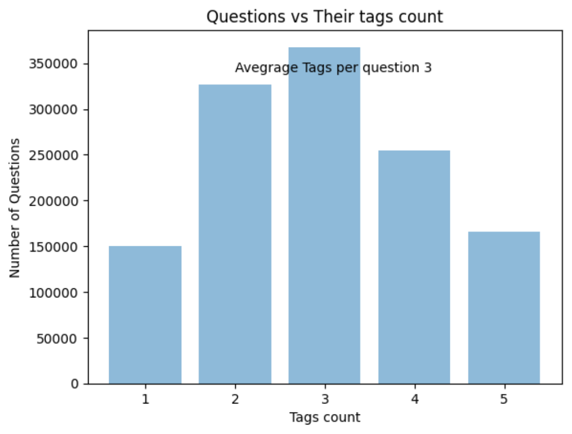

Fig 2: Average Tags per questions from EDA

Moreover, there were close to 14000 distinct tags linked to approximately 1.2 million diverse questions. Managing such a large number of tags for predication and classification posed a challenging task. To simplify this, we identified the 100 most frequent tags and exclusively employed them for the prediction task. This strategic removal of remaining tags further streamlined and downsized the dataset.

To assess the relationship between the body(content) of the question and its corresponding score, we initially eliminated HTML tags from the body text using the Python library BeautifulSoup. Additionally, we observed the minimum, maximum, and mean length of the body(content) text in the dataframe. The analysis revealed that the maximum body length was approximately 35000 characters, while the mean body length hovered around 1200 characters. It indicated a significant disparity between the maximum and mean body length, signifying. Some skewed data for were observed in EDA when we found maximum body length to be 35000 characters long and mean of body length to be 1200.

To address the skewed data issue, we constrained the body text length to five times the mean body length, resulting in a substantial reduction of data and rendering the training process less computationally intensive. Following these data cleaning steps, the newly created dataframe was prepared for subsequent preprocessing procedures.

C. *Preprocessing Pipeline*

To prepare the data for effective model training, it necessitates some form of preprocessing to optimize the model's performance. Below, we detail the steps conducted in the preprocessing pipeline we formulated:

Irrelevant raw HTML Tag: We eliminated irrelevant HTML tags using the Python library Beautiful soup from the raw data set.

To Lowercase: In NLP, words with the same meaning but having different letter cases are treated as distinct. To address this, all words were converted to lowercase during the input data feeding process. This ensures that the model considers words with the same meaning, regardless of letter case. For instance, words like "Query" and "query" were treated as identical.

Removal of Whitespace: Whitespaces occupy unnecessary empty space and do not contribute to information gain. Following the preceding steps, it was possible for the text to accumulate unnecessary whitespaces. To rectify this, whitespaces were removed, and single spaces were inserted in their place. This step aids in maintaining a clean and concise representation of the text data.

Punctuation Removal: Symbols and punctuation serve primarily to convey sentence structure and may or may not impact results depending on the application. In the case of this specific application, punctuation marks are not expected to significantly affect outcomes as StackOverflow questions are predominantly geared towards technical content rather than grammatical intricacies. Consequently, every punctuation symbols such as exclamation mark, colon, semicolon are eliminated from the provided text.

Tokenization: Tokenization is a method that entails dividing a provided text into distinct units. such as words or sentences, known as tokens. In our dataset, we applied word tokenization for further an analysis.

Lemmatization: It is a process that involves reducing words to their root forms or dictionary form, consolidating unique words in the text with similar inflections into a single form. With this reduction text can be processed as well as trained by the machine learning algorithm. For instance, term "high" shares its lemma with the word "higher".

Elimination of Stop Words Helping verbs, conjunction words or other auxiliary elements, in a text are deemed unnecessary for text analysis and categorises as stop words. These words, generally offer little value in text analysis. Their frequency in a text is often high, yet they do not significantly contribute to the model's ability to learn novel information. Common English stop words include 'a,' 'an,' 'was,' 'the,' 'is,' etc. It is important to note that these steps are vital for effective data preparation and efficient use in subsequent analyses.

D. *Feature Extraction and Data preparation*

After the cleaning of the dataset, text from the Title and Body columns is subjected to various preprocessing steps, and feature extraction is performed. During this process, several sequential steps are executed:

- First preprocessing step in data preparations is designed to trim the vocabulary size within the "Title" and "Body" columns of a dataset. Initially, the occurrence count of each unique word is computed using the "Counter" mechanism from the Python programming language's collection library. Following this, words that appear less than three times are excluded. This process is undertaken to streamline the representation of text data, emphasizing more prevalent and meaningful words while filtering out less significant terms that occur infrequently. This process aims to enhance performance by eliminating unnecessary words and reducing the dataset size, thereby contributing to data preparation.
- As a typical classification case now Title and Body serve as the independent variable, while their corresponding tags act as the label. Since each question is associated with multiple tags, addressing this prediction challenge involves a multi-label classification approach. To utilize these tags as output label and facilitate their input into classifiers, they must be converted into binary vectors. This conversion is achieved through the LabelPowerset and BinaryRelevance function from skmultilearn library.
- BinaryRelevance transforms a multi-label problem into multiple binary classification problems. It treats each label independently as a binary classification problem. Each label is considered separately, and a binary classifier is trained for each label.

After completing the aforementioned procedures, the data undergoes a transformation into TF-IDF vectors through tfidfvectorizer.

This method, a fundamental text embedding technique, converts Title, Body columns (input data) into embedding vectors. For this particular application, the TF-IDF vectors are configured with a maximum of 4000 features. Consequently, this generates input vectors with a length of 4000, subsequently utilized in training and prediction by the algorithms.

Ultimately, the integrated and stitched inputs columns i.e. Title and Body are fused through the utilization of the SciPy csr_matrix Python function. This amalgamated dataset will serve as the foundation for subsequent procedures and analyses.

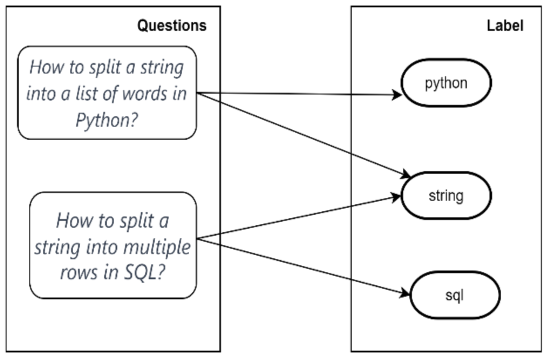
Questions
How to split a string into a list of words in Python?
How to split a string into multiple rows in SQL?
Label
python
string
sql

Fig 3: Common tag among unrelated questions

E. *Prediction model*

Following the completion of previous steps, prepared dataset is poised for model training. However, prior to this, it is essential to partition the data into training and testing sets. Leveraging the capabilities of the sklearn Python library, the dataset is split as 80 % training data and 20 % test data. Subsequently, a variety of classifiers models from the library are applied to this dataset.

SGD Classifier: Renowned for its effectiveness and efficiency on large-scale datasets, the SGD Classifier was employed for our classification task, yielding F1 score of .46 and precision of 81.16 percent.

Logistic Regression: Primarily used for classification tasks, logistic regression can be extended to multinomial regression for multi-class data. Given its strong performance, the Logistic Regression model achieved a precision of 80.10 percent and F1 score of .53.

Comparison of ML ALGORITHMS performance

CLASSIFIER	PRECISION	RECALL	F1 SCORE
Logistic Regression	0.8010	0.5338	0.6343
SGD Classifier	0.8116	0.4628	0.5773
Random Forest	0.8890	0.571	0.751
Linear SVC	0.798	0.689	0.738
Classifier chains	.832	.646	.819

Random Forest: In this study, the Python library Scikit-learn was employed to train a Random Forest classifier on the dataset, resulting in a precision of approximately F1 score of .619 and precision of 88 percent.
Classifier chains: In multilabel classification model, a set of binary classifier is used, where each classifier is responsible for predicting the absence and presence of specific label. Classifier chain resulted maximum 85 percent precision and F1 score of .43.

Above models demonstrated effective performance on dataset, despite its initial state not being directly suitable for the task. Additionally, to enhance prediction accuracy, we implemented a forwarfeeded neural network.
In following figure Feedforward Artificial Neural Network is described where Input Layer, dense hidden layer and SoftMax activation is shown.

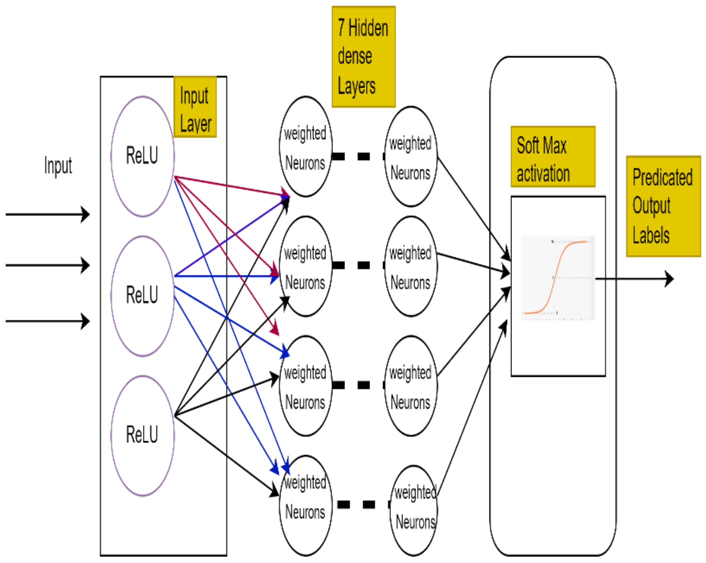

Fig 4: Feed Forword Artificial Neural Network

Input Layer has 7000 neurons, Other 8 Layers has 6000,4000,1100,600,400,200,175 and output layers has 200 Neurons w activation is Softmax activation
The final output layer utilizes the 'Softmax' as an activation function which can be described as

$$softmax(z) = \frac{e^{Zi}}{\sum_{j=1}^{N} e^{Zj}}$$

here, z⇒ Input to the Network functions

The implemented neural network consisted of 8 Dense hidden layers and 1 input layer, with each layer connected to the subsequent layer having a decreasing number of neurons as we progress through the network. Rectified Linear Unit 'ReLU' activation functions were applied to the hidden layers. The model underwent training on the provided dataset for 5 epochs and was able to acquire training accuracy of 70.07 percent and a testing accuracy of 58.34 percent.

In evaluating our models, we employed metrics such as Precision, Recall, and F1 score for each classifier.
Performance parameters utilized include Precision, F1 Score as well as recall.
Their definition can be simplified as follows

$$\text{Precision} = \frac{\text{True Postive}}{\text{False Positive+True Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{False Negative+True Positive}}$$

$$\text{F1 Score} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}$$

The Random Forest classifier exhibits the highest precision, followed by the Classifier Chains. However, considering

Upon analysing the neural network results, it is evident that the dense deep multilayered neural network exhibited considerably weaker performance compared to the classical machine learning models.

.

IV. CONCLUSION AND FUTURE WORK

This paper proposed an innovative and more efficient approach for the automatic question tagging task on Stack Overflow website's data. We used OneVsRest classifier and enabled binary classification algorithms such as SGD Classifier, and Regression model, and into multilabel classification.

The outcomes indicate noteworthy results, with the Random Forest achieving a precision of 88 percent. All the other models which were implemented demonstrate satisfactory performance. While there is potential for improvement in these algorithms for accuracy.

The performance of Neural network is also satisfactory with training accuracy of 70.07 percent and a testing accuracy of 58.34 percent.while nine layers of neurons with SoftMax activation were used.

We assert that dealing with the entirety of this question tagging dataset poses a substantial challenge due to its size and the associated computational expenses. In contemporary Natural Language Processing (NLP) practices, transformer architectures have gained popularity, particularly for addressing challenges related to long-range dependencies in sequential data like time-series and text data.

Furthermore, the integration of CNN, long short-term memory networks (LSTMs) and CNN with more hidden layer demonstrates substantial potential. These approaches may contribute to refining the Automatic Question Tagging problem further.

V. REFERENCES

[1] Robinson, Sara, and Yufeng Guo. "Predicting Stack Overflow Tags with Google's Cloud AI." Stack Overflow Blog, 22 July 2019, stackoverflow.blog/2019/05/06/predicting-stack-overflow-tags-with-googles-cloud-ai/.

[2]Saha, A. K., Saha, R. K., and Schneider, K. A. (2013). A discriminative model approach for suggesting tags automatically for Stack Overflow questions. 2013 10th Working Conference on Mining Software Repositories (MSR)

[3] V. Jain and J. Lodhavia, "Automatic Question Tagging using k-Nearest Neighbors and Random Forest," 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), 2020, pp. 1-4, doi: 10.1109/ISCV49265.2020.9204309

[4] Wang, X., Xia, X. Lo, D. TagCombine: Recommending Tags to Contents in Software Information Sites. J. Comput. Sci. Technol. 30, 1017–1035 (2015)

[5] S. Wang, D. Lo, B. Vasilescu, A. Serebrenik. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. ICSME, 2014.

[6] R. Miotto and C. Weng, "Unsupervised mining of frequent tags for clinical eligibility text indexing," J. Biomed. Inform., vol. 46, no. 6, pp. 1145–1151, 2013

[7] B. Sun, Y. Zhu, Y. Xiao, R. Xiao and Y. Wei," Automatic Question Tagging with Deep Neural Networks," in IEEE Transactions on Learning Technologies, vol. 12, no. 1, pp. 29-43, 1 Jan.-March 2019, doi: 10.1109/TLT.2018.2808187.

[8] Saini, T., and Tripathi, S. (2018). Predicting tags for stack overflow questions using different classifiers. 2018 4th International Conference on Recent Advances in Information Technology (RAIT).

[9] Rekha, V. Smrithi, N. Divya, and P. Sivakumar Bagavathi. "A hybrid auto-tagging system for stackoverflow forum questions." In Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing, pp. 1-5. 2014.

[10] Kavaler, David, et al. "Using and asking: Apis used in the android market and asked about in stackoverflow." Social Informatics. Springer International Publishing, 2013. 405-418.

[11] SD. E. Tas¸ar et al., "Auto-tagging of Short Conversational Sentences using Transformer Methods," 2021 Innovations in Intelligent Systems and Applications Conference (ASYU), 2021, pp. 1-6, doi: 10.1109/ASYU52992.2021.9598957.

[12] Anderson, Ashton, et al. "Discovering value from community activity on focused question answering sites: a case study of stack overflow." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012. [

[13] Stanley, Clayton, and Michael D. Byrne. "Predicting Tags for StackOverflow Posts." Proceedings of ICCM. 2013.

[14] Asaduzzaman, Muhammad, et al. "Answering questions about unanswered questions of stack overflow." Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013.

[15] Giménez, Jesús, and Lluis Marquez. "Fast and accurate partof-speech tagging: The SVM approach revisited." Recent Advances in Natural Language Processing III (2004): 153-162.

[16] Miltiadis Allamanis, Charles Sutton,"Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code.", MSR 2013, San Francisco, CA, USA

[17] Anusha S, Smrithi Rekha V, Bhagavathi Sivakumar P. "A Machine Learning Approach to Cluster the users of Stack Overflow Forum", ICAEES 2014, Tamilnadu, India