e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

AUTOMATED QUESTION TAGGING USING MACHINE LEARNING

Raja Ram A*1, Sanjay Kumar V*2, Ratheesh B*3

*1,2,3Department Of Computer Science Engineering, Panimalar Engineering College, India.

ABSTRACT

Stack Overflow is one among the foremost widely used platforms for asking questions and queries on topics associated with computing, software development and general programming. Tagging of the questions is especially useful for indexing information supported the tags. Currently, a user enters the tag manually for an issue asked by him/her. The question should contain a minimum of one tag manually typed by the user. It are often seen that the majority of the questions asked should either have more tags related to it or aren't tagged accurately and appropriately. Since there are an enormous number of tags, the method of rummaging through all the tags manually and find relevant ones are often cumbersome and is therefore overlooked by most of the users asking the questions. This research is concentrated on exploring methods for developing an autonomous tagging system using Machine learning methods like k-Nearest Neighbors and Random Forest alongside some crucial data preprocessing steps like Stemming, Tokenization and removing Stop words. The dataset for the above research is taken from kaggle.com which has a 10% Stack overflow question dataset open for all. The results of the subsequent proposed system for automatic tagging were satisfactory. Random Forest gave an average percentage accuracy of 70% across all the tags while k-Nearest Neighbors performed slightly better giving an accuracy of 75%.

I. INTRODUCTION

The task that we try to propose a solution for is that of tagging or labelling some given piece of text, specifically questions. With increasing number of users using the internet, there has been a tremendous growth of the Q&A websites. Q&A stands for Questions and Answers. Q&A sites like Quora, Stack overflow and forums of other non- Q&A sites receive a large number of questions each day. If the questions are not segregated or classified then they will get lost in the large pool of unanswered questions. For this reason, questions are classified by assigning them tags or labels. Assigning tags helps in clubbing similar questions. Also assigning proper tags will make it more likely for the question to reach the proper target audience and thus that question will have a higher chance to be answered [2]. One choice is to have employees to tag these questions. But there are several major drawbacks of such manual tagging. First of all, the scale of the Q&A websites is so large that it is practically infeasible for all of the questions to be tagged manually by employees. Secondly, even if we somehow overcome this then there is the issue of human bias that can lead to incorrect tagging of questions. Since manual tagging is so laborious and error-prone there is a need for a more robust solution. This is where automatic tagging comes in. In an automatic tagging system, we feed the question to the model that then assigns tags to the question. Internet has demonstrated itself as globally vital reason times Today, billions of people linked in internet by making proper use of that technology. This tagging technology can be used in abundant different ways to aid our comfort and ease . Tagging is a machine learning technique which provide tags to the information that user can easily identify the information they are looking for but there is flaw in that method most technology uses manual tagging and semi-manual tagging which is time consuming and people in that domain has the ability to identify the question and tag them it is not possible in real time and high in cost. We propose an automatic tagging method using NLP which automatically tag question where user can get what information he is searching to avoid shortcomings listed above which are manually tagged and high cost.

HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS

- Hard Disk : 80GB and Above
- RAM : 4GB and Above
- Processor : P IV and Above

SOFTWARE REQUIREMENTS

- Windows 7 or above
- JDK 1.7 and JDK 1.6

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

- Tomcat 6 and Tomcat 7
- Struts, Servlets

TECHNOLOGIES USED

- MACHINE LEARNING
- JAVA

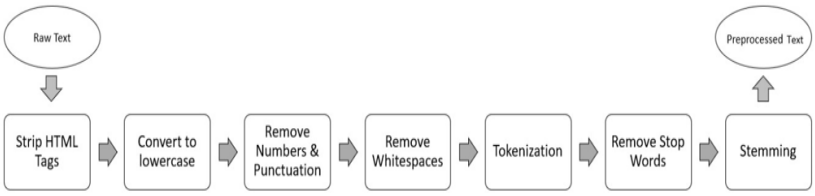
II. METHODOLOGY

Natural Language Processing(NLP)

Natural Language Processing (NLP) may be a collective term pertaining to automatic computational processing of human languages. This includes both algorithms that take human-produced text as input, and algorithms that produce natural looking text as outputs.

We propose our method for autonomous tagging system in this module. The data used to test our method was taken from "kaggle.com". The following steps illustrates our method in detail. The flowchart presented in this explains our proposed method in brief

Preprocessing Steps



A. Data Preprocessing

Before we could use the data to train our model, there was some preprocessing that needed to be done. In this section we describe the different stages of text preprocessing performed before extracting features to be used for training.

• **Strip HTML Tags**: The default text in the data is in HTML format. Since we would like to get rid of the tags as they do not necessarily add anything to understanding the context of the question. We use the Beautiful Soup library in Python for this.

• **Lowercase**: We then convert text to lowercase as case sensitivity doesn't add to the understanding of the question. Example "Python" and "python" both are equal and uppercase or lowercase letters don't make a difference to the question.

• **Remove Numbers**: While numbers can be useful in understanding some type of questions, for the most part they do not play any role in our task of determining the labels and so the numbers are removed from the text.

• **Remove Punctuation**: Punctuation too does not add any information to the context of the question and so the text is stripped of punctuation.

• **Remove Whitespaces**: After performing the previous steps, our text may end up having some unnecessary whitespaces. We thus remove these whitespaces and replace them with a single space.

• **Tokenization**: It is the process of breaking down of given text into smaller parts called "tokens". The tokens can be individual characters, words, sentences or even paragraphs. In our application, we use word tokenization in that we extract one word at a time from our text.

• **Remove stop words**: Stop words are words that do not add any information to the system and may even skew our text analysis. They are generally the frequently occur words in a language. For example, in English, some stop words could be "the", "a", "and", etc. We discuss feature extraction in detail in the next section where we use the most frequent 100 words as features for a tag. If we didn't remove the stop words, then due to their high frequency they would end up being the features for every tag

• **Stemming:** It is the process of reducing the words to their root, or base word. It can be seen as removing a suffix (possibly empty) from the word, or equivalently, replacing a word with one of its non-empty prefixes. It is done in order to reduce the similar words that convey the same information to equivalent terms thereby reducing the size of feature space. For example, the words "want", "wants", "wanted" are essentially the same for understanding the context of a text and so the stemmer might decide to reduce each one of them to the

word “wan”. After successfully applying these preprocessing steps on the question title and body, we can then move on to extract features for different tags.

B. Feature Extraction

This is one of the most important steps in building the automatic tagging system. The feature extraction process from the question takes place after the data-preprocessing step. The questions in the database were converted to words by following the pre-processing steps. The questions were first converted to units of words, also known as tokenization. After tokenization, stop words like, “in” ,” as”, ”the”, ”and”, ”a”, ”an” were removed as they are insignificant to the question and don’t carry much information as such. The final step is Stemming and we used Porter Stemmer to stem the remaining words. Feature extraction is then done as follows:

• First, top hundred tags by frequency are extracted from the database. These are the most frequently appearing tags that are present in most of the stack overflow questions.

• For each of these hundred tags, we select thousand questions (positive examples) and eight-hundred negative examples to have a balance between positive and negative examples. This is done as SVM (Support vector Machine) performs poorly on unbalanced data as cited in [4] and thus, to remove the bias, we provide the algorithm with a good balance of both negative and positive examples.

• We just chose the thousand positive examples for each tag and extract the top hundred frequently occur words in the questions. For all hundred tags, there may be many overlapping words but on the worst case the dimension of our feature vector will be (n*10,000), where n is the number of data rows which the model will be trained on.

C. Prediction Models

We used the classifiers from python’s scikit-learn library, more specifically, we used k-Nearest Neighbors (KNN) and Random Forest (RF) algorithms on our data set.

K-Nearest Neighbors (KNN)

We use a KNN (k-Nearest Neighbors) Classifier for our classification task. It is a simple and comes under supervised machine learning algorithm. It can also be used for Regression problems but we use it for Classification. We explain the KNN algorithm in this section. KNN is a supervised learning algorithm and it requires labelled data. The basis on which the algorithm relies is as follows: similarly classified data points will be close to each other in the feature space that is, they will be clustered together. When we need to classify a new data point, we find its ‘k’ nearest neighbors, where ‘k’ is a parameter of the model. The new data point is then assigned the class label of that class which appears for the maximum number of times in those ‘k’ nearest neighbors. For example, if k=5, and the 5 nearest neighbors for a data point are 0, 0, 1, 0, 1 then the class of 0 will be assigned to the point. For KNN to work well, the data should support its basis, that is, the data should follow the assumption that similarly classified data points are closer to each other. To define this ‘closeness’, or similarity between two data points, we use the distance measure in the feature space. More specifically, a point is closest to another if the Euclidean distance between them is lesser compared to all other points in the dataset. The Euclidean distance is measured in the feature space, thus in our case, with 100 features, it will be measured in a 100- dimensional space with each feature acting as a dimension. In our dataset, we use frequently appearing words for a tag as our features. Thus, in our case many similarly classified questions will have similar values for the features as frequently appearing words are more likely to appear in positive examples and less likely to appear in negative examples. We also find that from our results that KNN provides a satisfactory result. The parameter of the model is ‘k’. For our application, we used the value of k=5. We found that the results were satisfactory as well as computationally efficient. The table-1 indicates the percentage accuracy of k-NN on the top five tags (sorted by frequency) on the train and test data. As we expected, JavaScript is the most frequent tag followed by java.

Random Forest

We also used Random Forest Classifier for our classification task. We tuned the maximum depth parameter in a range of 3 to 8 and found that at maximum depth = 5, the Random Forest classifier performed the best. The Random Forest Classifier gave us an accuracy of 72%. We explain the Random Forest Classifier as follows: The Random Forest is an ensemble machine learning method, which uses a number of decision trees to predict/classify the given data. We are using python’s famous library ‘sci-kit learn’ where the random forest for

the dataset can be built efficiently. The hyper parameters for building the random forest were tested multiple and the below mentioned hyper parameters gave the most promising results:

• n estimators = 150 (Determines the number of trees in the forest).

• max depth = 5 (Determines the maximum depth of each decision tree). Random forest doesn't just average the results of all decision trees. The two important concepts used to build the forest are:

• Random Sampling of training dataset

• Random features considered when splitting the node

III. SYSTEM ANALYSIS

EXISTING SYSTEM

In existing system user search different information in internet to get the knowledge about the content in this environment there is possibility that user can't get the desired information or it's hard to seek out the specified information or can't get the clear knowledge about it and doesn't give any alternative solution to the user these methods are through with manual tagging and semi-automatic tagging First manually tagging questions with knowledge units needs that the taggers be specialist therein subject an issue bank usually contains an enormous number of questions, which are updated constantly this makes manual tagging costly in terms of the time taken and related cost and Semi-automatic tagging analyzes content and returns tags that require to be more processed by users, making human help mandatory. There is high chance that questions are often mismatched.

Problem Definition

- Manual tagging is time consuming and needs well manual attention.
- Semi-automatic tagging the content has to be again processed.
- High possibility can't receive requested user query questions
- High cost and time consuming.

PROPOSED SYSTEM

We propose an automatic tagging of questions by using NLP (Natural language processing) one of the machine learning techniques. Machine Learning (ML) is applications of artificial intelligence (AI) that provides systems the ability to automatically learn. NLP analyzes understand the human language in a smart and useful way. The previous systems are build using manual and semi-auto tag which isn't suitable in making daily updates of knowledge and isn't possible to get requested information and mines the unwanted answers (information) to the user that are confusing and tags are mismatched wrong it makes difficult to observe the information. In this work we done an automatic question tagging that overcome all existing problems. The Medical datasets are collected from Medinet library. The data are stored within the CSV file formats for the later use. CSV file contains the questions and answers from the medical health domain and then the automatic tagging takes place of the question and Answer then the users search the query in AWS (Amazon Web Services) and the query is forwarded to the Medinet Library to extract the user queried related answer. Medinet library contains 700 medical domain files which is stored in "tc2011 "API. If the requested file not found in Medinet, will redirect it to PDF box it will provide pdf files which contains information about domain or diseases related to the question. The pdf files are extracted by using Lucene indexing which give fast retrieval of files. If the answer is not related to query, it will be forwarded to the Expert(doctor) who can clear the doubts and replay for the query. And finally providing the quiz for the student who want know their knowledge in the medical field and providing them domain score and overall score and feedback for the student.

Advantages

- Automatic tagging does not need help from human
- Provide standard and consistent results
- Low cost
- Easily retrieves the answer tagging makes easier to understand the information
- Provides Knowledge for the students

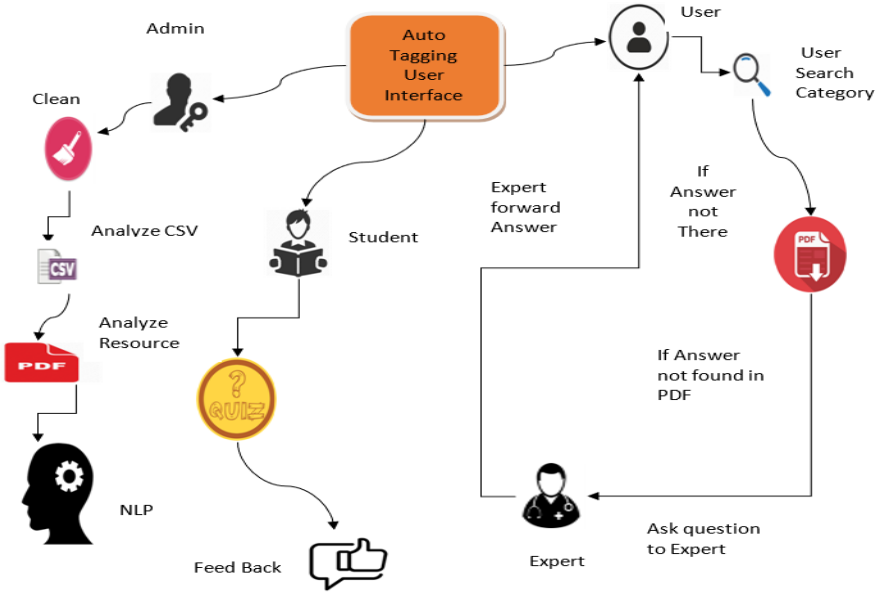
SYSTEM DESIGN

First module Admin has to register first and then provide admin details (Name and password) valid admin enters the page. Admin is responsible for the whole operation admin work includes cleaning, adding the CSV files, analyzing resources, NLP (Natural language processing) and cleaning NLP. The user has to register their details first after that user login using (Name and Password) if user is valid, then user enters the page and chooses the categories of domain present in the page user clicks the categories it show list available domains then user select any one domain based on that related information or question and answers to that domain will be displayed. If the requested domain information or question answer is not available it provides PDFs related to that domain the provided answers of the question are not satisfied when user feels. User can ask question to the Expert (doctor) who can clearly answer the user query. Expert (doctor) has to register first like (Name, Expert id, category of doctor) after that login using valid credentials. Here expert id is unique id for the expert and category is to identify the expert domain. Expert enters the page and he gets the notification from the user who asked the question after that Expert analyze the question and provides the related answer to the question the answer is forwarded to the user profile. User can view the answer any time, student registers first after the login page is forwarded to the quiz page the student is asked to write quiz in medical domain. Questions from medical domain is provided student has to answer it finally student gets the overall score of the test, domain score, and gets the feedback for the test which the student can improve knowledge in domain which he choose by knowing the overall score

System Features

We propose models for automatic question tagging, an important approach to question management that has received little attention. The mechanism proposed to capture important information from questions can be applied to other tasks such as question text and short text mining.

Overall System Architecture



MODULES USED

1.Admin Preprocessing

2. Auto tagging Questions

3. Expert's Answering Process

4. Student Assessment

MODULES EXPLANATION

1. **Admin Preprocessing**

First module Admin has to register first and then provide admin details (Name and password) valid admin enters the page. Admin is liable for the entire operation admin work includes cleaning, adding the CSV files, analyzing resources, NLP (Natural language processing) and cleaning NLP.

2. **Auto tagging Questions**

In this module the user has to register their details first after that user login using (Name and Password) if user is valid, user enters the page and chooses the categories of domain present in the page user clicks the categories it show list available domains then user select any one domain based on that related information or question and answers to that domain will be displayed. If the requested domain information or question answer isn't available it provides PDFs associated with that domain the provided answers of the question aren't satisfied when user feels. User can ask question to the Expert (doctor) who can clearly answer the user query.

3. **Expert's Answering Process**

In this module Expert (doctor) has got to register first like (Name, Expert id, category of doctor) then login using valid credentials. Here expert id is unique id for the expert and category is to identify the expert domain. Expert enters the page and he gets the notification from the user who asked the question after that Expert analyze the question and provides the related answer to the question the solution is forwarded to the user profile. User can view the answer any time,

4. **Student Assessment**

In this module student registers first after the login page is forwarded to the quiz page the student is asked to write quiz in medical domain. Questions from medical domain is provided student has got to answer it finally student gets the general score of the test, domain score, and gets the feedback for the test which the student can improve knowledge in that domain.

IV. RESULTS AND CONCLUSION

This paper proposed an efficient method for autonomous tagging of questions for forum websites such as stack overflow. The obtained results are quite good with Random Forest classifier giving an accuracy of 70% while the K-Nearest Neighbor algorithm giving an accuracy of 75%. We have used traditional Machine Learning Classification algorithms namely K-Nearest Neighbor (KNN) and Random Forest (RF).The main aim is to give knowledge regarding Health and diseases of human by receiving user query was implemented in this machine learning based java web application

FUTURE SCOPE

While they have provided satisfactory results, there is always scope for improvement.. Both the industry and research community are moving towards using Deep Learning and Neural Networks for modern problems instead of traditional Machine Learning Algorithms. While there has been some work in Automatic Question Tagging using Neural Networks, the majority of work has been focused on using traditional algorithms like Support Vector Machine (SVM) and Naive Bayes as we saw in the previous work section. . We believe that Deep Neural Networks have great potential and can thus be used for the task of Automatic Question tagging.

ACKNOWLEDGEMENTS

My heartful thanks to my project guide Professor Mr. M. Shanmuganathan who actually helped us in accepting our project and provide some guidance to our project research work.

V. REFERENCES

[1] A. Al-Hmouz, J. Shen, R. Al-Hmouz, and J. Yan, "Modeling and simulation of an adaptive neuro-fuzzy inference system (anfis) for mobile learning," IEEE Trans. Learn. Technol., vol. 5, no. 3, pp. 226– 237, 2012.

[2] P. Brusilovsky, "Knowledgetree": A distributed architecture for adaptive e-learning," in Proc. 13th Int. World Wide Web Conf. Alternate Track Papers & Posters. ACM, 2004, pp. 104–113.

[3] M. Mendicino, L. Razzaq, and N. T. Heffernan, "A comparison of traditional homework to computer-supported homework," J. Res. on Technol. in Edu., vol. 41, no. 3, pp. 331–359, 2009.

[4] J. D. Forbey and Y. S. Ben-Porath, "Computerized adaptive personality testing: a review and illustration with the mmpi-2 computerized adaptive version." Psychological Assessment, vol. 19, no. 1, p. 14, 2007.