Tagging and classification of stack overflow questions

Helena Julie Arpudaraj
University of Central Florida
helena_julie@knights.ucf.edu

Kripalakshmi Babu
University of Central Florida
b.kripalakshmi@knights.ucf.edu

Ebin Scaria
University of Central Florida
ebin@knights.ucf.edu

ABSTRACT

To provide better experience for the users, tagging is encouraged in questions and answers websites like Quora, Askville, stack overflow. Manual tagging is cumbersome. Users often find it difficult to search and enter the correct tag. It will be helpful to have auto-tagging which will detect the correct tag and suggest the user based on the question they have asked. In this paper, we are going to present auto-tagging and classification of stack overflow questions by feature extraction and building multi-label classification model which tags questions and classifies them into multiple groups based on the tags.

KEYWORDS

Auto-tagging, multi-label classification

1 INTRODUCTION

Leading questions and answers discussion websites like stack overflow, Quora, stack exchange, Askville are growing very fast. The number of questions and answers in these forums are exponentially increasing and it would be very difficult for the users to find the answers. For better interaction, it is important to tag the user asked questions on various topics. The tagging will give better answers related to the question asked by the user and the previous answers given by the other users. There are some forums which will allow us to manually enter the tag which could be a hassle for the users. To make it more convenient and user-friendly experience automatic tagging is introduced, this type of grouping and tagging the questions will give the users easier and quicker responses for the questions they are interested in, which makes the forums more interesting and more responsive to the users.

Grouping of questions allows users to provide quick answers to questions they are interested in responding to. Currently, users are asked to tag the questions based on their understanding however this is cumbersome as users are unable to tag the questions properly and unsure about the most suitable tags[22]. Through this project, we aim to automatically predict, tag and classify the questions with the most suitable tags. The questions on stack overflow are asked by many different users and the manual tags are entered by the users for the quick response by the users. Since there are so many tags entered by the users, not all tags are relevant. This causes questions to be directed towards the wrong group of people and thus the questions being unanswered.

Figure 1: An example of stack overflow question with manually entered tags

The fig 1 depicts an example of a stack overflow question which contains many manually entered tags by the users for quicker response. The goal of the project is to extract feature words and build a multi-label classification model that tags the questions and classifies them into multiple groups based on tags. We are using a dataset that we prepared by aggregating the questions and tags from various other stack Overflow datasets. We have represented our data in n-gram, tf-idf, and Word2Vec format. We will build models that have been adapted to multi-label classification tasks like One Vs Rest linear SVM, Multilearn adaptive kNN and Naïve Bayes. Performance comparison of the models with respect to the different word representations will be provided at the end of the project.

The main motive of the paper is to extract features using n-gram, tf-idf, and Word2Vec and automatically tag by using multi-label classification models such as SVM OneVsRest, Naïve Bayes and MLkNN. The evaluation metrics used are subset accuracy, Hamming loss, Jaccard similarity coefficient, precision, and F1 score.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The remaining paper is organized as follows: in section 2 and 3, we present the related work and background for our project. In section 4, we present the methodology we use in our work. In section 5, the evaluation metrics and results of our work are reported. In section 6 and section 7, we will discuss the shortcomings, improvisations, future work and conclusion. Section 8 gives the work distribution of how we distributed work within the team and section 9 has the link to our code for this project.

2 RELATED WORK

Tags can be of two types: in-text keyword and out-of-text keyword. The in-text keyword is based on keywords in contents. The out-of-text keyword will have a tag that is maintained outside the contents. Tagging is useful in a web search when user wants questions related to a specific keyword. The user will type the keyword and all the posts with that tag are retrieved. Hence, auto-tagging is important and tags must represent semantics or meaning of the post [24].

The paper [14] proposes an automatic tag recommendation algorithm called TagCombine. There are three components of TagCombine- predict tags using a multi-label learning algorithm, the similarity-based ranking which involves similar objects to recommend tags and tag-term based ranking which involves historical affinity of tags to certain words. [14]. The TagCombine is used to recommend tags in software info sites. The tags in the software info sites are first investigated and then the tags are recommended. The tags are even recommended for untagged objects by checking similar tags. Linear combination method is used to tune the parameters.

To avoid tag explosion phenomenon, [18] proposed machine learning based method which could do automatic mapping for the user entered random tags to their identical Wikipedia concept. Precision, recall and F1 scores are evaluated in the paper [18].

[23] proposed a system which can automatically predict the tags and assign them to the questions in the stack overflow forum. The tags are assigned based on the context in the question. They used SVM using content-based classifier and programming language detection system. It is a simple classifier which will automatically predict the tags based on body and title.

[25] used stack overflow dataset because it is constrained and contains many posts and tags are associated and developed tag prediction system Bayesian probabilistic model which can predict the hashtags which were already used by the author. If the best tags are not present for the question, this model could predict the best tag associated with the question which will automatically increase the possibility to get the best answer for the question. This model can also be used as a tag cleanup system, where the tags for the already existing questions are compared. If the tag activation is low, the tag is either removed or considered for further analysis.

2.1 Data Pre-processing

In the paper [2], they have proposed ontology-based auto-tagging of out-of-text keywords. They have also discussed the pre-processing steps and tagging. Pre-processing steps involves data preparation for tagging. The tagging process involves the classification and selection of tags. In the pre-processing step, they create the term-weight matrix that gives the TF-IDF weight of terms. This term weight matrix is then used for the classification.

The TF/IDF weight of terms is given by:

$$TF\_IDFweight_{i,j} = tf_i \times IDF_i$$

where $TF\_IDFweight_{i,j}$ is a TF-IDF weight of the term i in the domain j, tf_i is term frequency of the term i in the articles of the domain j, $IDF_i = Log\frac{D}{df_i}$. D is the number of the domain of the train dataset df_i is the number of the domains that have the term i. [24].

2.2 Feature Extraction

Feature extraction is currently done by TF-IDF(Term Frequency-Inverse Document Frequency) and Word2Vec. We refer to the paper "Autonomous Tagging of Stack Overflow Questions" [15] [14] which uses the TF-IDF method (Term Frequency-Inverse Document Frequency) for feature selection. The Word2Vec method is another process which gives a set of feature vectors[3]. These methods help us in evaluating the importance of each word in the corpus. [7][13] [20].

3 BACKGROUND

3.1 Word cloud

Word could is a famous data visualization technique which is related to textual data. The tags used are generally single words and will be of different size and color. This method is generally used to focus important, famous and trending words based on the frequency in which they are used. These more frequent words are displayed in a bigger and bolder size. They are also called as tag clouds or text clouds.

3.2 Term frequency

Term frequency determines how often a word repeats in a document. Length of the document is one of the major parts in finding the frequency of a term. The frequency of a word in a longer document might be high compared to the shorter document. Thus, the length of the document plays a vital role in determining the term frequency. To determine the term frequency (TF), The number of times a word t is present

in a document is divided by the length of the document i.e, the total number of terms in the document.

$$TF(t) = \frac{(Number\ of\ times\ word\ t\ present\ in\ a\ document)}{(Length\ of\ the\ document)}$$

3.3 TF-IDF

Term frequency-inverse document frequency(TF-IDF) measures the importance of a word in a document in the whole corpus. Term frequency (TF) measures the importance of the word in a document. We want to pick the most frequently used words and also at the same time do not want to pick words like 'the' and 'this' which occurs frequently in all documents. For this we use Inverse Document Frequency(IDF).[1]

$$TF(t) = \frac{(Total\ occurrence\ of\ word\ t\ in\ document)}{(Total\ number\ of\ words\ in\ the\ document)}$$

$$IDF(t) = Log\frac{(Number\ of\ documents\ in\ the\ corpus}{(Number\ of\ documents\ with\ word\ t)}$$

3.4 Word2Vec

Word2Vec is a probability-based representation of words. The context of the word, semantic and syntactic relation with other words is captured in Word2Vec. This model contains both skip-gram which predicts the target word from the context and CBOW which predicts the context from the target word. These are shallow neural networks.

3.5 Bag of words

Bag of words (BOW) is a method which extracts features from text documents and creates feature vectors. These feature vectors can be used in Machine Learning algorithms. A vocabulary is created for all the unique terms in the document. It can also be defined as a method, which is the collection of terms to represent a sentence along with its word count [8].

3.6 Multi-label Classification

The aim of the project is to perform a multi-label classification instead of multi-class classification. By using NLTK and sklearn, the stop-words present in the tags are restored and the rest of the stop-words were removed. In multi-label classification, we implemented a one vs rest classification using Support Vector Machines with a Linear kernel, Naïve Bayes and Multilabel k Nearest Neighbours(MLkNN).

Another technique is to use ensemble methods, in which the voting scheme is used for deciding the multi-labels after multiclass classifications. [5] explains a very popular method of multi-label classification using OneVsRest classifier in Linear SVC using Scikit-Learn. This is similar to the binary relevance method [15][14]. The paper "Autonomous Tagging of Stack Overflow Questions" [5] also proposes multi-label classification using SVM OneVsRest classifiers.

(1) Random Forest: Random Forest is a supervised learning model. Random Forest uses multiple Decision Trees [19]. It groups the results of various decision trees to get a more accurate and stable prediction. Random forest is used for classification and regression problems. Random Forest also solves overfitting issues. Since there are more trees it avoids overfitting [27].
(2) ML-kNN: Multilabel kNN is used to detect multi-label data, It is derived from the tradition kNN algorithm and it is a lazy algorithm. For an unknown term, the k nearest neighbor in the training set are detected. Depending on the information obtained from the neighboring data, the label for the unknown data is found.
(3) Naïve Bayes: It is an efficient algorithm but it should be adapted to multi-label data. In multi-label learning, OneVsRest strategy can be used, in which the multiple labels, for instance, are predicted by the classifier. For multi-label classification, Naïve Bayes must be wrapped in OneVsRest classifier
(4) Support Vector Machine: It is a supervised learning model. It is used for both classification and regression problems. An SVM builds a model using the training data and this model is used to classify test data. In SVM we can improve the performance by selecting different kernels parameter and the margin parameter C. It avoids overfitting. It is robust as it maximizes margin [14][26][21].
(5) Adaboost: It is an ensemble classifier. It's a boosting algorithm which combines weak classifiers to create a strong classifier. It uses two or more classifier algorithms to increase the accuracy. Adaboost can be used for binary as well as multiclass classification [17] .
(6) Logistic Regression: Logistic regression is used when the dependent(target) variables are categorical. Logistic regression is given by following expression

$$P(y = 1|x) = \frac{1}{1 + exp(-x)}$$

Hypothesis

$$Z = WX + Bh\Theta(x)$$

When Z goes to infinity, Y will become 1. When Z goes to negative infinity, Y will become 0. We can decide a threshold to decide the class of Y.

Decision boundary can be linear, non-linear and polynomial. Data is fit into the linear regression model, the logistic function is used to predict the target categorical dependent variable [9].

In the paper [14] they generate a Document-term Matrix in the pre-processing step. The matrix represents the frequencies of occurrence of words in a document/post. This was done using bag-of-words. Also called as 1-gram. They have removed stop words. Words with a frequency less than 2 percent were removed [16]. After this, they partition the matrix into training and test data by dividing the matrix to 70-30 percent training and test data. Then they pass the training data into SVM and random forest functions to train the model.

4 METHODOLOGY

The fig 2 depicts the overall pipeline for our model. The raw text data is given as the input and the next step is preprocessing. Data analysis and data visualization are done using word cloud and tf-idf analysis. Then we represent the text as feature vectors using Word2Vec, BOW(Bag of words) and tf-idf and then send to multi-label classifiers like OneVsRest SVC, MLkNN, and multi-label Naïve Bayes classifier.

4.0.1 Data collection.

(1) We had initially decided to use the Stack Lite dataset from Kaggle for the purpose of this project[6]. But unfortunately, we ran into a lot of issues when we started using the dataset. The size of the dataset was very small, and It did not have enough textual information to apply any of the natural language processing techniques. So, we prepared a dataset by aggregating stack overflow questions and tags form various online sources and publicly available datasets. After preparing the dataset, we analyzed the data to gain some insights. We found that the data had over a hundred thousand questions and 1768 tags. We realized that it was not possible to perform classification on such a large dataset due to computational restrictions. We then took measures to reduce the size of the dataset. The dataset now has twenty-five thousand records.

4.0.2 Feature Selection.

(1) We analyzed the entire tag space using a word cloud to identify the most important tags and identified the 20 most important tags. We confirmed our results using tf-idf and removed all the questions that did not belong to the top 20 tags. The next challenge we faced was that fact that after deleting the less frequent tags the data set became imbalanced most of the questions belonged to 9 tags and the remaining questions belonged the other 11 tags. We balanced the dataset by adding more questions to the dataset. However, we were not able to balance the dataset completely as we were unable to find tags for all the questions.

4.0.3 Data Prepossessing.

(1) The data from stack overflow is extremely unstructured due to the diverse set of users who post questions. We removed all links, program snippets, special characters and punctuation from the question and body for all the rows. Finally, we removed all the stop words and tokenized the sentences. In the next step, we lemmatized the words using NLTK in python. Since the title had the important tokens, we copied the title thrice and concatenated it with the body.

4.0.4 Feature Extraction and Representation.

(1) **BOW (bag of words):** We used tf-idf to extract feature words. We want to pick the most frequently used words and also at the same time do not want to pick words like 'the' and 'this' which occurs frequently in all questions. So we used tf and idf(TF-IDF) to extract feature words. We used TfidfTransformer of sklearn to create tf-idf values. Now we have the extracted features/keywords for each question in the dataset [4]. Then we used BOW(bag of words) to create vectors. We used CountVectorizer in sklearn for this and also used it to remove words that appeared more than 90 percent of the question.

(2) **Word2Vec:** In this step we used the tokens of each sentence in the corpus to generate word embeddings using Word2Vec. We used the Word2Vec function from the gensim library. The dimensionality of words was set to 256. We tried both CBOW and skip gram approach. We got slightly better results with skip grams. After we generated the word embeddings we trained our Word2Vec model on the corpus. We then converted the embeddings to vectors so that we can use them for classification. We took the average of all the word vectors in the sentence and this average represents the sentences. Another approach that was used was that we multiplied the word vectors with their tf-idf weights and then took the average of all the words in the sentence to form a vector that represents the sentence.

4.0.5 Classification.

(1) For multi-label classification we implemented a one vs rest classification using Support Vector machines

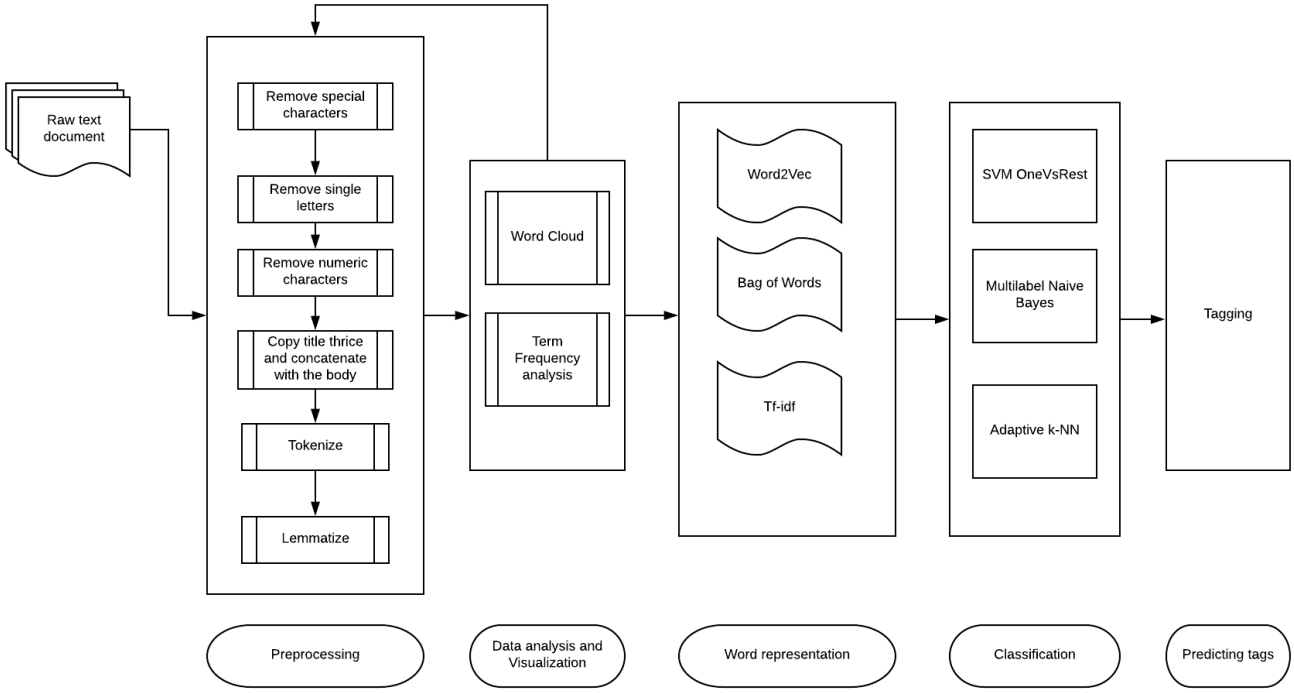

Figure 2: Pipeline for our model

with a Linear kernel, Naïve Bayes and Multilabel k Nearest Neighbours(MLkNN).

Initially, we got very bad results. The model was not able to predict the tags. So we went back and tried to analyze. We found out that the reason for it was the mistakes in the pre-processing stage. When we removed all the stop words from the dataset, we had overlooked the fact that some of these stop words were actually words that were part of the tags. For example, single letter like a,c,d, etc and words like at, dot, etc. We performed pre-processing on the corpus again using NLTK and sklearn. This time we kept all the stop words that were part of the tags and deleted the rest of the stop words. We achieved this by editing the stop words file that NLTK and sklearn uses.

After this, the model was working pretty well in predicting the tags. But we found out that the subset accuracy was very less. We analyzed the dataset and found that the dataset had tags given by the users too. The users made many mistakes in the tags while tagging their questions. For example, the users used wrong spellings or tags that were not very useful. Our model was not predicting such user entered tags and were only predicting the tags from feature vectors.

In our project, we used OneVsRest SVC classifier. It uses ensemble methods, in which the voting scheme is used for deciding the multi-labels after multiclass classifications. We used the same approach used in [5]. It explains a very popular method of multi-label classification using OneVsRest classifier in Linear SVC using Scikit-Learn.

We used BinaryRelevance from scikit-multilearn for multilabel classification using Naïve Bayes. This transforms the multilabel problem to binary classification problems. It will independently train each label as one binary classifier.[10]

For ML-kNN we use adapted kNN for multi-label classification. We used skmultilearn.adapt.MLkNN in python for this purpose.

5 EVALUATION

5.1 Dataset

The StackLite dataset was initially chosen for the project the same as [6]. This dataset contains the questions and tags which were posted in the stackoverflow.com by the user who got the login account and it is free to post in the stackoverflow.com forum. Id, title, body, and tags were the column

present in the StackLite dataset. The questions were related to the programming language and it contained so many links. Since the questions were too small with insufficient data, we ran into a lot of issues when we used the dataset. The insufficient data made it difficult to apply machine learning techniques in the initial StackLite dataset.

We prepared a dataset by collecting stack overflow questions and tags form various online sources and publicly available datasets. It contains tags, title, body and Id columns. It contains a mechanical and electrical question. After preparing the dataset, we analyzed the data to gain some insights. We found that the data had over four hundred thousand questions and 1768 tags. We realized that it was not possible to perform classification on such a large dataset due to computational restrictions. We then took measures to reduce the size of the dataset. The data collected from the stack overflow was unstructured because the questions and tags were posted by the users with some human error. It also contains hyperlinks, program snippets, punctuation, and special characters. We removed all these and also the unwanted stopwords and restored the rest of the stopwords present in tags.

5.2 Evaluation metrics

In our project, as it's a classification model we try to evaluate our model results using the below terms

- Subset accuracy: It indicates the percentage of correctly labeled samples. Since it's a multi-label problem we used subset accuracy. The subset accuracy calculates the accuracy based on the exact match between the set of labels in the predicted result and the actual result.

$$Exact\ Match\ Ratio, MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

I denotes the indicator function

The main disadvantage of this measure is that in multi-label problems might contain partially correct labels and we ignore those partially correct results here. In sklearn *accuracy_score* is used for subset accuracy.[11]

- Hamming loss: It is the fraction of label not correctly predicted to the total number of labels predicted.

$$\frac{i}{|N|.|L|}\sum_{i=1}^{|N|}\sum_{j=1}^{|L|} xor(y_{i,j}, z_{i,j})$$

Here $y_{i,j}$ is the target and $z_{i,j}$ is the prediction [2]. N is the total number of instances, L is the number of labels and xor is the logical equality of $y_{i,j}, z_{i,j}$

- Jaccard similarity coefficient: It is the measure of similarity between the predicted results and the actual results in the test dataset. Mathematically, it can be represented as the cardinality of the intersection of two sets divided by cardinality of the union of two sets. In other words, for a pair of data, the element of pair occurs on the set on entities where the calculation will be performed [12]. A higher Jaccard similarity coefficient represents a higher similarity between the predicted results and the actual results in the test dataset.

$$J(A, B) = \frac{\text{mod}\ (X \cap Y)}{\text{mod}\ (X \cup Y)}$$

here X and Y correlate to the set of entities which occur with A and B

- Precision: It explains the proportion of positive identifications that was predicted right.

$$Precision = \frac{TP}{(TP + FP)}$$

.

here TP is True positive value and FP is false positive value

- F1 Score: It shows the balance between the precision and the recall

$$F1\ score = 2 * \frac{(precision * recall)}{(precision + recall)}$$

These evaluations will be done for multi-label classifications using OneVsRest SVC, kNN and Naïve Bayes.

5.3 Evaluation results

```
 Air Seal recess can without attic access
Air Seal recess can without attic access
Air Seal recess can without attic accessI
recently discover moisture attic mold
inspector recommend seal can light house
We 30 vault ceiling attic access   n nAny
tip seal fan below  n nFor reference
House Seattle  build 2008  Canned light IC
air tite halo models  I surprise leak air
air tite  But I imagine give seal like
bathroom recess light closet recess light
glass cover gasket   n nIf I take baffle
unit metal screw such   n nDo I caulk
Sheetrock around open form tape crevice
use duct tape  n => attic, insulation,
recessed
```

Figure 3: Example of Tag prediction for a StackLite question

The fig 3 depicts an example of a prediction of important tags for a stack overflow question. There are three important tags predicted for the given question by the user. The fig 4 shows the evaluation results obtained for OneVsRest SVC, Naïve Bayes and Multilabel kNN using word embeddings from BOW and TF-IDF. These are evaluated and compared based on the subset accuracy, hamming loss, Jaccard similarity coefficient, precision, and F1 score.

	Subset accuracy	Hamming Loss	Jaccard Similarity Ratio	Precision	F1 score
OneVsRest SVC	0.1266	0.0028	0.3557	0.2708	0.1936
Naïve Bayes	0.1121	0.0026	0.3427	0.2718	0.2009
MLkNN	0.1301	0.0026	0.3555	0.2998	0.1925

Figure 4: Evaluation Results

The fig 4 shows the output obtained for OneVsRest SVC. The accuracy is comparatively high in this classification. The Hamming loss and the Jaccard similarity coefficient is also high compared to the other two classifiers. The overall subset accuracy is higher for multi-label kNN. The precision is also higher. The hamming loss is comparatively less in MLkNN and Naïve Bayes.

Our Word2Vec model wasn't giving us good results. It wasn't giving us good predictions. The subset accuracy being 0.008, Jaccard similarity coefficient being 0.02 and hamming loss being 0.003. We are still trying to improve our Word2Vec model.

The tags in our dataset also had manually entered tags by the users. As there are so many unnecessary tags entered by the users, there are some errors. These errors in user entered tags is one of the reasons for our low subset accuracy. Our model predicts tags based on the word embeddings and features extracted. Hence, the model was not able to predict the tags users entered manually. We will try to work on this in the future and remove the error tags that are causing low subset accuracy and precision.

6 DISCUSSION

We presented a simple model which automatically classifies and predicts stack overflow tags. Memory was one of the major constraints as our initial dataset was too large and the size of the question, i.e, body, and title column were too large. The final dataset we use in our work is stack overflow dataset from kaggle. This dataset contains tags, body, title columns. The tags in this dataset are manually entered by the user for their quicker and better response. As there are so many unnecessary tags entered by the users, there are some errors. These errors in user entered tags were the reason for our low subset accuracy.

Our model predicts tags based on the word embeddings and features extracted. Hence, the model was not able to predict the tags users entered manually. We will try to improve our evaluation results by removing the error tags while calculating the accuracy. Evaluation metrics are different for multi-class and multi-label classification. For multi-class, we can use accuracy as an evaluation metric and in multi-label, we used subset accuracy where we calculate accuracy based on the exact match between the set of labels in the predicted result and the actual result. We used the evaluation metrics like hamming loss, subset accuracy, Jaccard similarity coefficient, F1 score and precision in our project. We found that the subset accuracy is very low because only the important tags for each question is predicted and all other unnecessary tags are not predicted.

7 CONCLUSION

In this paper, we used the multi-label classifier to classify and automatically predict tags for the user given questions. We predicted the most important tags for the questions. Along with finding the important tags, our initial aim was to find all the tags associated with the questions. Our future work can be expanded by auto-tagging many tags for the given question for quicker response. The subset accuracy and performance can still be improved. The run time can also be optimized.

8 WORK DISTRIBUTION

(1) Helena Julie Arpudaraj - PID 4735186
- Check for better datasets and data cleaning.
- TF and removing stop words.
- BOW and TF-IDF representation coding.
- One Vs Rest SVC coding and evaluation.
- Naïve Bayes coding and evaluation.
- Similarity analysis.
- Report writing.
- Evaluation metrics.
- Performance analysis and suggesting improvements.
- Presentation.

(2) Kripalakshmi Babu Venkateswaran - PID 4541508
- Dataset Collection and cleaning.
- Word cloud visualization using python.
- Multi-learn kNN classification
- Report writing.
- Evaluation metrics.
- Performance analysis and suggesting improvements
- Presentation.

(3) Ebin Scaria - PID 4447183
- Dataset Collection and cleaning.
- Data pre-processing.
- Word cloud visualization using python.
- BOW and TF-IDF representation coding.
- Word2Vec representation coding.
- One Vs Rest SVC coding evaluation.
- Evaluation metrics.
- Performance analysis and suggesting improvements
- Coding.
- Presentation.

9 OTHER RESOURCES

This is the link to our code for this project, Code link

REFERENCES

[1] 2010. Term frequency-inverse document frequency. http://www.tfidf.com/

[2] 2015. multi-label classification. https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

[3] 2017. Classification and Auto-Tagging of Stack Exchange Questions. https://github.com/AAbercrombie0492/nlp_final_project

[4] 2017. feature extraction tf-idf. http://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.XIHENChKg2x

[5] 2017. Solving Multi-Label Classification problems. https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/

[6] 2017. StackLite. https://www.kaggle.com/stackoverflow/stacklite

[7] 2018. Auto Tagging Stack Overflow Questions. https://towardsdatascience.com/auto-tagging-stack-overflow-questions-5426af692904

[8] 2018. Bag of Words. https://medium.freecodecamp.org/an-introduction-to-bag-of-words

[9] 2018. Logistic Regression. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[10] 2019. BinaryRelevance. https://en.wikipedia.org/wiki/Multi-label_classification

[11] 2019. Multilabel evaluation metrics. https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

[12] Jacob Bank and Benjamin Cole. 2008. Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia. *Wikipedia Similarity Team* (2008), 1–18.

[13] André CPLF de Carvalho and Alex A Freitas. 2009. A tutorial on multi-label classification techniques. In *Foundations of Computational Intelligence Volume 5*. Springer, 177–195.

[14] M Eric, A Klimovic, and V Zhong. 2014. # ML# NLP: Autonomous Tagging of Stack Overflow Questions.

[15] Mihail Eric, Ana Klimovic, and Victor Zhong. 2014. {meric, anakli, vzhong}@ stanford. edu December 8, 2014. (2014).

[16] Ingo Feinerer. 2013. Introduction to the tm Package Text Mining in R. *Accessible en ligne: http://cran. r-project. org/web/packages/tm/vignettes/tm. pdf* (2013).

[17] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface* 2, 3 (2009), 349–360.

[18] Arash Joorabchi, Michael English, and Abdulhussain E Mahdi. 2015. Automatic mapping of user tags to Wikipedia concepts: The case of a Q&A website–StackOverflow. *Journal of Information Science* 41, 5 (2015), 570–583.

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

[20] Rohithkumar Nagulapati, Mayanka Chandrashekar, and Yugyung Lee. 2018. Transformation from Publications to Diabetes Ontology using Topic-based Assertion Discovery. In *2018 IEEE International Conference on Healthcare Informatics Workshop (ICHI-W)*. IEEE, 9–18.

[21] S Nashat, Azizi Abdullah, S Aramvith, and MZ Abdullah. 2011. Support vector machine approach to real-time inspection of biscuits on moving conveyor belt. *Computers and Electronics in Agriculture* 75, 1 (2011), 147–158.

[22] V Smrithi Rekha, N Divya, and P Sivakumar Bagavathi. 2014. A hybrid auto-tagging system for stackoverflow forum questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*. ACM, 56.

[23] Sebastian Schuster, Wanying Zhu, and Yiying Cheng. [n. d.]. Predicting tags for stackoverflow questions. ([n. d.]).

[24] Gridaphat Sriharee. 2015. An ontology-based approach to auto-tagging articles. *Vietnam Journal of Computer Science* 2, 2 (2015), 85–94.

[25] Clayton Stanley and Michael D Byrne. [n. d.]. Predicting tags for stackoverflow posts.

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

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