TagSuggest: A Stack Overflow Tag Recommendation System using One-Vs-Rest Multi-label Classification

Trixia R. Belleza and Concepcion L. Khan

***Abstract*— Stack Overflow aims to organize questions by requiring users to categorize their posts with tags. However, this tagging process is done manually by users which can cause inaccurate classification of posts and disruption of the website′s overall user-experience. Hence, the objective of this study is to recommend tags based on the title and question content. The three classifiers, Support Vector Machine(SVM), Logistic Regression(LR), and Multinomial Naive Bayes(MNB), were applied in a one-vs-rest approach and were measured by accuracy and f1 score. Among the three, SVM attained the highest overall performance having an accuracy of 93.39186% and an f1 score of 77.07187%. Since SVM produced the best performance among the three, it was then used as a classifier to the tag recommendation chrome extension application. A system usability scale (SUS) was used to evaluate the usability, complexity, and learnability of the application. TagSuggest got an overall SUS score of 74.4079 which is considered to be above average.**

***Index Terms*— supervised learning, support vector machine, logistic regression, multinomial naive bayes, text-classification, one-vs-rest, multi-label classifier, chrome extension, system usability score**

I. Introduction

A. Background of the Study

A large population of users use online question-and-answer forums, like Stack Overflow and Quora, as a platform for discussing common topics of interest [1].

Unlike Quora, Stack Overflow is mainly for computer science majors and programmers [2]. The website is also strict with its standards for user interaction. In fact, according to the Stack Overflow website (2018) [3], questions are organized into well-defined categories based on its content in a form of tags. Tags are words that describe the question content in order to connect people that are knowledgeable in that field. With this, users can easily filter-search for questions.

In related studies, Guan et al [4] used a graph-based approach in dealing with tag-recommendation systems, in which, tags are being chosen according to their relevance to the content and preference of the user using a graph-based ranking algorithm. In addition, collaborative tagging data was used as an input to their algorithm.

However, in this study, the problem has been taken from another perspective and a different strategy. Instead of using collaborative tagging, the questions posted on the website will be used as input data.

Generally, using questions as the dataset, the machine will be able to classify them based on their topic category.

B. Statement of the Problem

Tagging of questions are done manually by a user, which disrupts the website′s user-experience [5]. In fact, the researchers, Short, Wong, and Zeng (2014) [6], mentioned that some inexperienced users are having a hard time identifying which category their question belongs to. Moreover, manual tagging is prone to human error, which may result to improperly tagged questions. Hence, this can form a disarray of information that could overturn its purpose.

C. Significance of the Study

The burden of manual tagging of questions on Stack Overflow calls for the need of an automated text-classification machine.

In order to classify texts, a multi-label classification technique, called the one-vs-rest approach, will be used in this study. This is especially used in classifying text-content to multiple categories according to Liu and Chen (2015) [7].

D. Objectives of the Study

Generally, the study aims to create an application that recommends tags to Stack Overflow questions using a multi-label classifier. Specifically, it aims:

1) To train and test twenty thousand data from Stack Overflow using Support Vector Machine(SVM), Logistic Regression(LR), and Multinomial Naive Bayes(MNB);
2) to evaluate the performance of the classifiers by computing accuracy, precision, recall, and f1 score with a 60% f1 score threshold in classifying questions according to their topic categories;
3) to create a chrome extension that recommends tags to Stack Overflow questions using the model with the highest overall performance among SVM, LR, and MNB; and
4) to evaluate the application to at least 30 users using a System Usability Scale and obtain an overall above average score (68 and above).

E. Scope and Limitation of the Study

The language of the text should be in English and the actual data labels that were used in this study are limited to Python, JavaScript, Java, C, R, MySQL, HTML, CSS, and control flow statements such as while-loop, for-loop, and if-statement.

II. Review of Related Literature

Stack Overflow, an online forum, has been widely used for discovering new technologies and analyzing topics in computer science. In this website, posts are organized according to their topic category. In order to achieve this, users must manually classify categories that describe the question they are posting. However, manual tagging of questions degrades the website′s user experience which led researchers to improve the website′s features.

A. Stack Overflow: A Question and Answer Forum

Stack Overflow is an online question-and-answer forum especially made for programmers. It is a website for discovering new trends and technologies through the discussions of its users [2]. According to Short, Wong, and Zheng (2014) [6], the website offers a platform for users to discuss about the question posted by another user. In posting a question, the user must provide the title and the body of the question including tags as a means for describing what the topic is all about. These tags are essential in providing filtered searches and organized posts.

However, manual tagging of posts can be quite tedious, which is inefficient in terms of user-experience [5].

B. Machine Learning

Machine Learning is the science of getting computers to detect patterns from a trend, and use it to predict a possible outcome using computational methods.

One type of machine learning is supervised learning. Supervised learning is a method in which target labels are known. In other words, there is a desired output based on a given input [8]. This type of machine learning is commonly used in automated text classification given a set of labels [9]. To demonstrate, in order for the machine to classify a sample, it should be given a set of labeled data to work on first. Then, the machine will be given new unlabeled texts for the testing phase, in which the machine will automatically classify them based on the model it has learned [10].

According to Alpaydin (2010) [11], the results of machine learning algorithms can become reliable for analysts, data scientists, and businessmen to create decisions and uncover hidden insights. With this, machine learning has been widely used because of its many applications. One application would be the automatic detection of acromegaly, a hormonal disorder, from facial photographs [12]. Another would be an automatic classification of app reviews which is essential for app developers to improve their app [13].

Machine learning has been the cause of changes in people′s day-to-day activities and it still continues to prove its potential up to this day.

C. Natural Language Processing Toolkit

The natural language processing toolkit (NLTK) is a most commonly used platform by researchers, teachers, and programmers. It is a suite of libraries and programs for classification, tokenization, stemming, parsing, etc., especially made for processing the human language [14].

Over the years, NLTK has continued to improve with more complex data structures and processing tasks [15].

D. Multi-label Classification

A multi-label classification assigns an instance to multiple target labels. In the study of Wang, Chen, and Suis (2018) [16], multi-label classification was applied in order to assign texts to multiple categories based on conceptual content. Thus, the training set contains the texts associated with a set of categories. The goal is to predict the categories of unseen text data based on the model created from the training set.

Another study was conducted by Almeida et al [17] using multi-label classification as a means of analyzing the sentiments found in text. In fact, this methodology was able to categorize multiple emotions in the same text.

It is often confused with the definition of multi-class classification. To differentiate, a multi-class classification is mutually exclusive, wherein, each instance is only mapped to one label. In a multi-label classification, two or more labels can be assigned to each instance.

According to Nooney (2018) [18], ”Most traditional learning algorithms are developed for single-label classification problems.” Because of this, solutions to multi-label classification problems are transformed into multiple single-label problems, so that the existing single-label algorithms can be used.

According to Boutell et al in 2004 (as cited by Tsoumakas & Katakis, n.d.) [19], two of the techniques in multi-label classification decomposes the problem into multiple single-label classification problems. One is the one-vs-rest approach where samples that are associated with the label are positive and the rest, which represent other labels, are negative. In other words, this approach distinguishes one class from the rest of the other classes [20]. On the other hand, the one-vs-one approach trains each sample with only two classes in which, it learns to distinguish each pair of classes [21].

E. Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm that is based on the concept of decision planes. Decision planes are the boundaries that separates classes of objects. According to Gholami and Fakhari (2017) [22], the goal of the SVM is to find a decision function which can classify unseen data with a good accuracy.

Decision planes classify data linearly. However, most classification problems are not always linearly separable and more often, complex structures are needed in order to separate one class from the other [24].

To solve this problem, a kernel machine was introduced. The idea is that, the data that is not linearly separable in a two-dimensional space may be linearly separable in a higher dimensional space by computing the inner products of the data points [25].

Support Vector Machines have been widely used in text-classification systems because of its outstanding classification accuracy compared to other classification algorithms [26].

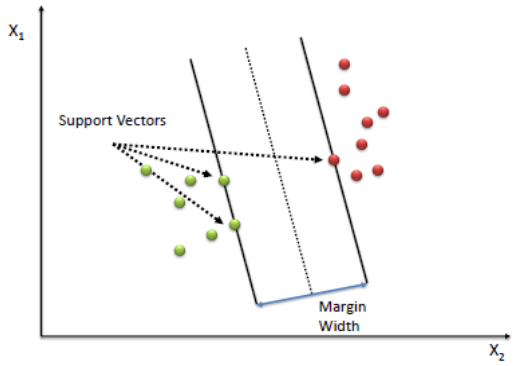

Fig. 1. An SVM that classifies two classes using a linear kernel in two dimensions [23]

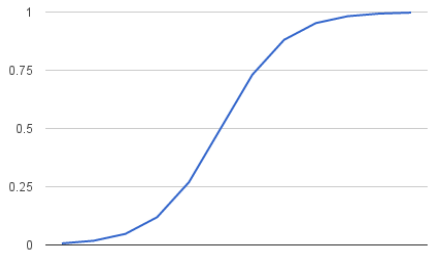

Fig. 2. Graph of a logistic function [34]

F. Naive Bayes

Naive Bayes algorithm is a classification technique that uses the probability theory called Bayes theorem. The formula for the Bayes' theorem is as shown below where h is the hypothesis and d is the data.

$$P(h|d) = \frac{P(d|h) * P(h)}{P(d)}$$

Conditional probabilities are basically based on prior knowledge of conditions that are related to the event. Bayes theorem states that if h and d represent two events, the probability of h given d, P(h|d) denotes the conditional probability of h occurring, given that d occurs. On the other hand, the probability of d given h, P(d|h), denotes the probability of d occurring, given that h occurs. In addition to Bayes theorem, it states that these two conditional probabilities are related to each other [27].

One of the event models of Naive Bayes is the Multinomial Naive Bayes model which takes into account the frequency of the terms in a document and solves for the conditional probability of a term given a category [28]

To illustrate how Naive Bayes algorithm works, Techopedia (2018) [29] uses sorting of fruits as an example. A classifier that categorizes fruits into apples and oranges would use shape and color as input features but would not assume all these things at once since apples and oranges have a similar round shape.

According to Wei et al (2011) [30], Naive Bayes algorithm has an advantage for multi-label classification problem. In their study, they mentioned that McCallum [31] used this approach to classify multi-labeled documents and Zhang et al [32] used the same algorithm in dealing with multi-label problems that incorporated feature selection mechanisms that gave the machine learning technique a higher accuracy.

G. Logistic Regression

Logistic regression is a statistical analysis technique which is used when the dependent variable is dichotomous, where 1 represents the value of TRUE and 0 denotes FALSE [33].

In order to define logistic regression more clearly, Saedsayad [23], a website that explains machine learning algorithms, compared linear regression from logistic regression based on what the linear regression cannot do that the logistic regression can. It was stated that a linear regression has a large range of values that goes outside the acceptable range 0 to 1. Since binary or dichotomous datasets are limited to two possible values only, the residuals will not be normally distributed on the predicted line.

Unlike linear regression, logistic regression forms a logistic curve that limits the values from 0 to 1. The curve is formed using the natural logarithm of the odds of the target labels.

In 2016, Jason Brownlee [34] described the logistic regression using the graph on fig 2.

He added that input values(x) and weights such as the bias or intercept term(b0) and coefficients of x(b1) are combined in order to predict an output value(y). The coefficients of x are estimated from the training data.

$$y = \frac{e^{b0+b1x}}{1 + e^{b0+b1x}}$$

In related studies, Fujino and Isozaki (2008) [35] used logistic regression for a patent mining task. The logistic regression models were trained by providing existing patent documents as dataset. They designed the feature vectors of the documents with weighting and component selection methods to avoid overfitting.

H. Grid Search

Grid search is the process of tuning the hyperparameters of a model by scanning the data then builds a model using the optimal parameters it has found [36]. In the process, Grid Search also evaluates the model for each combination of the hyperparameter values. This is essential it bringing out the best in a model based on the hyperparameter values that were tuned [37].

I. Tag Recommendation System

Tagging is a form of categorizing texts using keywords to describe its content. This mechanism has become popular on the Web because it promises to provide organization and easy retrieval of online content [38].

According to Guan et al [4], a tag recommendation system is necessary to help users decide tags more accurately by

suggesting suitable and relevant tags to them. It can also enrich the index of resources.

Moreover, the focus of their study lies on the documents that are annotated by users. With tags, users can bookmark the Web URLs they visited.

The researchers used a collaborative filtering strategy that explores collaborative knowledge and not on the content of the documents.

However, considering the diverse interests of users, the performance of the tag recommendation system provided a low-level accuracy in finding tags directly related to the current document.

In another study, the tag recommendation system can be modeled by focusing on the most frequent tags assigned to a text-content and the tagging history of the user, in which, tag recommendation algorithms can be classified into user-centered and resource-centered ones [39].

Meanwhile, Schuster, Zhu, and Cheng developed a tag recommendation system which computes a feature vector for each document-label pair which resulted to a much slower performance. They suggested that it would have been faster if the one-vs-rest strategy was applied which trains a model for each label [40].

III. Materials and Methods

A. General Flowchart

The flow of the study is shown below:

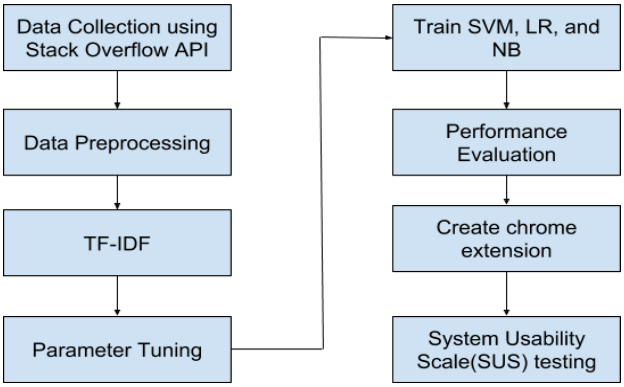

Fig. 3. A flowchart showing the summary of the process.

B. Frameworks and Libraries

In creating the backend of the application, Python3, together with the libraries sklearn (Scikit-Learn), nltk (Natural Language Processing Toolkit), pymysql, pandas, and pickle were used.

The sklearn library was used for model selection and feature extraction. It was also used in measuring the accuracy, precision, recall, and f-score of a classifier. The nltk library was used in stemming. Pymysql was used in creating a MySQL and Python connection. Dataframes were created after loading the data from MySQL using the Pandas library and Pickle was used for saving and loading the models. A python server was also created using the library Flask.

In creating the chrome extension application, JavaScript was used. With AJAX, a connection between the Python server and the JavaScript code was established.

C. Data Collection

Stack Overflow provides an Application Programming Interface (API) that outputs a JSON-format list of questions with tags.

The question body from the Stack Overflow API contained HTML tags which deemed insignificant in the study. Thus, these HTML tags were removed using the html2text library before inserting the data into MySQL.

Pymysql library was used in inserting the scraped data to MySQL.

D. Data Preprocessing

The language of the text-data that was used in this study was English.

In text data, there are multiple variants of a root word. Prefix, suffix, or infix are usually present in a word which express the same meaning with its root word. Thus, stemming was used to reduce words to their corresponding root words, such as ”retrieval”, ”retrieved”, ”retrieves” will be considered as ”retrieve”. A Natural Language Toolkit (NLTK) library was used in stemming the text data. All text data were also converted in lower-case form to make it case-insensitive to the program.

The actual data labels that were used in this study are the eight commonly used tags in Stack Overflow. These are JavaScript, Java, C, R, MySQL, HTML, CSS and Python [3]. Control flow statements, namely, if-statement, for-loop, and while-loop, were also used. Thus, the total number of actual data labels for this study was eleven.

E. Bag-of-Words Vocabulary Model

The bag-of-words model is a representation of data in natural language processing. It contains the set of words, excluding stop words, and the number of times the word has occurred [34].

F. Term Frequency-Inverse Document Frequency (TF-IDF)

According to Erra et al (2013) [41], the Term Frequency-Inverse Document Frequency (TF-IDF) provides the weights of a word depending on its significance to a text. TF-IDF is basically the product of Term Frequency (TF) and Inverse Document Frequency (IDF). TF is computed by

$$tf(t,d) = \frac{f_{t,d}}{max\{f'_{t,d} : t' \in d\}}$$

TF is the frequency f of the term t in the document d.
On the other hand, **IDF** is computed by

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|}$$

Where N is total number of documents and $|d \in D : t \in d|$ is the number of documents d in which the term t appears.

If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |d \in D : t \in d|$.

Finally, TF-IDF can be computed by multiplying TF and IDF.

$$tf(t,d) * idf(t,d,D) = tf(t,d) \times idf(t,D)$$

G. K-Fold Cross Validation

The data set was divided into k groups (or folds) of equal size. In this study, the value of k was ten. The first fold is the validation set and the method is fit on the remaining k-1 folds.

The training data set contains the questions from Stack Overflow and the tags of the questions. The test data set was used to test for accuracy, precision, recall, and f-score by giving the machine a set of data without labels.

H. Parameter Tuning

To improve the accuracy of the classifiers, hyperparameters were tuned by setting different values and choosing among those values which provided the best score.

In order to choose the best value for a parameter, K-Fold cross validation was used together with the Grid Search Algorithm which returns the value that has achieved the best score.

I. Training the Model using a Multi-label Classifier

In this study, the researcher used a one-vs-rest multi-label classifier approach. To classify the N-class data in a one-vs-rest approach, N binary classifiers were created, $\{a_1, a_2, \ldots a_N\}$. For the i-th classifier, let all the points in the class a_i be one, and let all the points not in the class a_i be zero [5].

Basically, multi-label classification outputs a model that maps x features to y binary vectors by assigning a binary value, 0 or 1, for each label in y. Then, a machine learning algorithm was applied which trained each model that was produced by the one-vs-rest classifier.

Three machine learning algorithms, SVM, MNB, and LR, were applied to the one-vs-rest classifier and were evaluated as to which among them has provided the highest accuracy, precision, recall, and f1 score.

J. Performance Evaluation

According to Lingras and Butz (2007) [42], to measure the performance of the classifier, the researcher will be using three metrics: precision, recall, and F-score.

Precision is computed by finding the ratio of correctly classified samples to the total classified positive samples.

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

Recall is computed by finding the ratio of correctly classified samples to the total samples in the actual class.

$$Recall = \frac{TP}{TP + FN}$$

Finally, to compute for F-score that will determine the accuracy of a classifier is as follows:

$$Fscore = \frac{2 * Precision * Recall}{Precision + Recall}$$

In addition to the performance evaluation metric, the accuracy scores of the models were also measured. In multilabel classification, the accuracy score is computed based on the set of labels of a sample that exactly match the true corresponding set of labels.

K. Stack Overflow Chrome Extension

In order to apply this study to the Stack Overflow website, the researcher created a chrome extension.

In this study, Python was used for the main program while JavaScript was used in creating a chrome extension. Hence, the researcher created an AJAX call to connect the Python Flask server to JavaScript.

L. System Usabilty Scale

The System Usability Scale(SUS) is an evaluation technique wherein testers or users choose a degree from 1 to 5, 1 being the lowest and 5 being the highest, based on the statement provided. The statements test the learnability, and the usability of the application [43].

To test the chrome extension application, SUS was used for alpha testers to give their sentiments and suggestions based on the overall user-experience of the application.

IV. Results and Discussions

The twenty thousand dataset that was gathered from the Stack Overflow API was divided into three sets: training set (60% of the total dataset), validation set (20% of the total dataset), and test set (20% of the total dataset). Initially, the model was fit on a training dataset which contained the input text data and its corresponding target label. Successively, the trained model was applied to the validation dataset in order to tune the model's hyperparameters (e.g. the c value in a support vector machine) using the ten-fold cross validation. Finally, the test dataset was used to evaluate the final model using the hyperparameters that attained the highest f1 score during the training period.

After evaluation, a chrome extension application was then created in order to apply it to the Stack Overflow website. A system usability scale was also used for users to evaluate the application based on its usability, learnability, and consistency in recommending tags.

A. Performance Evaluation

*1) **Training Evaluation**:*

Table I. Tuned Parameters of SVM for Each Label

Labels	C	tol	Training F1 Score
Python	1	1	85.39807
JS	1	1	83.10246
Java	2	1	89.39346
C	1	0.001	91.51906
R	1	0.01	89.79137
MySQL	2	1	88.41865
HTML	1	1	80.40604
If	1	0.01	76.52010
While	2	1	71.95958
For	1	1	68.83868
CSS	1	0.001	85.31148

Table I shows the values of the hyperparameters of SVM that achieved the best f1 score in a ten-fold cross validation using the Grid Search algorithm.

The second column in table I denotes the hyperparameter C which is essential in structual risk minimization to avoid overfitting. It influences the length of the margin hyperplane that separates the two classes. On the other hand, the third column pertains to the hyperparameter tol, short for tolerance. This tells the model to stop searching for a minimum (or maximum) once a certain tolerance is achieved.

Table II. Tuned Parameters of LR for Each Label

Labels	C	tol	Training F1 Score
Python	4	0.001	83.82296
JS	4	0.0001	82.28695
Java	4	1	89.55012
C	4	1	91.15534
R	4	0.01	88.75614
MySQL	3	1	87.98429
HTML	4	0.01	80.04841
If	4	0.001	75.52289
While	4	0.01	70.49201
For	4	1	68.12207
CSS	4	0.01	84.74140

Similarly, table II shows the values of the hyperparameters of LR that attained the best f1 score in a ten-fold cross validation using the same algorithm that was used in table I.

As was the case in SVM, the second column pertains to the hyperparameter C which denotes the regularization of the model. Although this does not improve the performance of the classifier on the training dataset, it can improve the generalization performance on unseen data.

The third column shows the tolerance value of the model per label that acts like the tol hyperparameter of SVM.

Table III. Tuned Parameters of MNB for Each Label

Labels	Alpha	Training F1 Score
Python	0.1	82.69168
JS	0.01	74.91571
Java	0.01	88.49700
C	0.01	90.67926
R	0.01	89.03908
MySQL	0.1	86.19440
HTML	0.1	80.90462
If	0.01	69.01341
While	0.1	71.60526
For	0.1	67.90615
CSS	0.01	85.30064

Table III shows the alpha values of MNB for each label. This is an additive Laplace smoothing to solve a frequency-based probability of zero.

Since this is a one-vs-rest multilabel classification technique, the hyperparameters of each classifier were tuned in order to achieve the best model for each category.

*2) **Testing Evaluation**:* Applying the hyperparameters that achieved the best f1 scores, the accuracy, precision, recall, and f1 score were measured using the unseen dataset or test dataset. A ten-fold cross validation was used in order to see the consistency of our model in different folds.

a) Accuracy:

Table IV. Accuracy of SVM, LR, and MNB for Each Fold

Folds	SVM	LR	MNB
Fold1	94.29145	93.01516	91.39505
Fold2	90.48184	90.7192	89.06955
Fold3	92.92666	92.00973	91.18206
Fold4	96.17855	95.85806	95.26466
Fold5	94.89675	94.94422	94.13719
Fold6	94.13719	93.95917	93.22336
Fold7	92.89105	92.55875	91.02777
Fold8	90.79041	90.67173	89.74602
Fold9	93.59127	93.75742	92.33325
Fold10	93.73368	94.24401	92.74864
Average	93.39186	93.173745	92.01276

Table IV shows the accuracy of the three classifiers for each fold. Accuracy is the ratio of the correct predictions over the total number of predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

This means that on average, the Support Vector Machine (SVM) classifier has the highest percentage of getting correct predictions, which is 93.39186%. The Logistic Regression (LR) classifier provided a higher percentage than the Multinomial Naive Bayes (MNB) classifier, which were 93.173745% and 92.01276% respectively.

b) Precision:

Table V. Precision of SVM, LR, and MNB for Each Fold

Folds	SVM	LR	MNB
Fold1	84.61533	84.00063	79.33919
Fold2	82.95278	82.32081	78.86721
Fold3	79.61223	78.57505	74.66497
Fold4	76.45867	77.48253	74.66605
Fold5	77.54195	76.90954	75.53958
Fold6	77.13012	77.09699	74.74759
Fold7	80.89948	80.02398	75.96483
Fold8	81.83367	81.31577	78.04727
Fold9	74.82407	75.57103	71.05133
Fold10	79.59055	81.54296	77.93633
Average	79.54588	79.48392	76.08244

Precision measures the fraction that were correctly predicted positives out of all the samples the classifier had labeled positive.

$$precision = \frac{TP}{TP + FP} = \frac{TP}{TotalPositivesfromClassifier}$$

Based on table V, the SVM classifier has the highest proportion of positive predictions that were actually correct. On average, the SVM classifier is correct 79.54588% of the time, the LR classifier is correct 79.48392% of the time. Meanwhile, the MNB classifier provided a precision of 76.08244%.

c) Recall:

Table VI. Recall of SVM, LR, and MNB for Each Fold

Folds	SVM	LR	MNB
Fold1	78.04966	77.77755	74.18626
Fold2	76.06528	75.44681	75.59499
Fold3	75.60958	73.81056	72.34078
Fold4	75.46245	73.84916	73.80373
Fold5	76.08108	75.76621	76.04837
Fold6	78.05407	76.85093	72.91093
Fold7	78.29167	79.69424	75.81492
Fold8	80.63994	80.08633	77.90670
Fold9	77.42761	77.19784	77.63890
Fold10	79.02518	79.86368	77.80880
Average	77.47065	77.03433	75.40544

Recall is the proportion of correctly predicted positives out of the total actual positives.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{TotalActualPositives}$$

It was shown on table VV that on average, SVM correctly identifies 77.47065% out of its actual labels, while the LR and MNB classifier correctly identifies 77.03433% and 75.40544% out of their corresponding actual labels respectively.

d) F1 Score:

Table VII. F1 score of SVM, LR, and MNB for Each Fold

Folds	SVM	LR	MNB
Fold1	80.53242	80.01726	76.05602
Fold2	77.83075	77.49900	75.93159
Fold3	76.38291	75.79731	72.57042
Fold4	75.47811	75.17813	73.07770
Fold5	75.42768	75.57231	74.49878
Fold6	75.28034	74.23199	73.35897
Fold7	78.54478	78.13206	74.94811
Fold8	78.95318	78.89959	76.08247
Fold9	74.44274	74.34303	71.91470
Fold10	77.84583	79.66128	74.47295
Average	77.07187	76.93320	74.47295

F1 score provides the balance between the precision and recall. In the multi-label case, F1 score is computed by the average of the F1 scores of each class.

$$f1score = \frac{2 * precision * recall}{precision + recall}$$

On average, the SVM provided the highest F1 score, which is 77.07187% , LR with 76.93320% F1 score and MNB with 74.47295% F1 score.

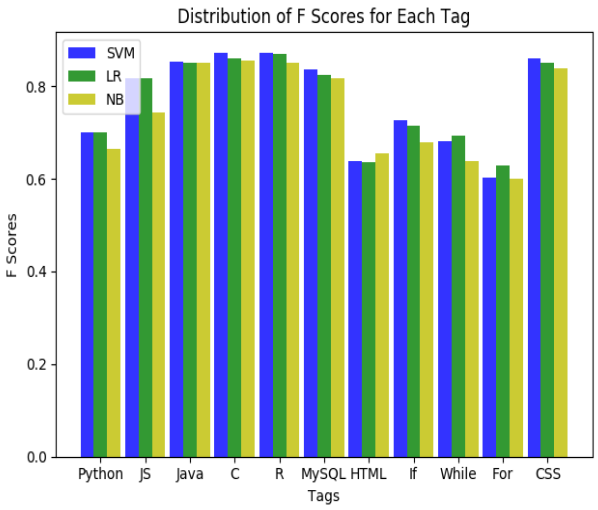

Fig. 4. A bar graph showing the f1 scores of SVM, LR, and NB by tag.

In the bar graph shown in fig 4, the SVC (Support Vector Classifier) or SVM attained the highest f1 score for all categories. The LR classifier's f1 scores differ only by a small amount from SVM.

B. Visualization of the Application

Figure 5 and 6 shows the interface of the classifier applied to the Stack Overflow website in which the user inputs a title and a question then clicks the "Tag Suggestions" button in order to show the tag recommendations that are produced by the classifier.

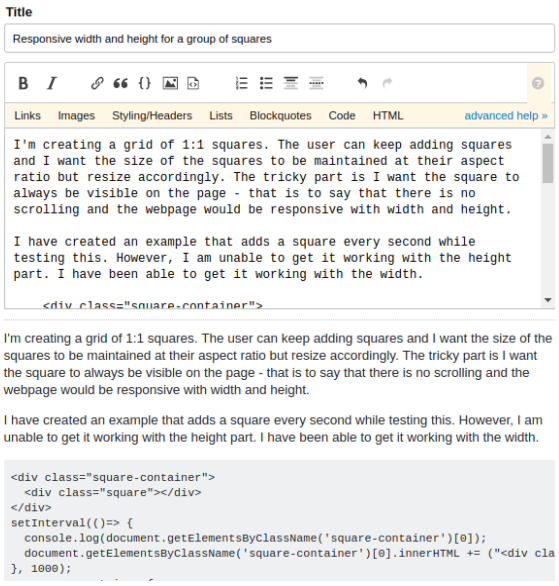

Fig. 5. User Interface of Stack Overflow.

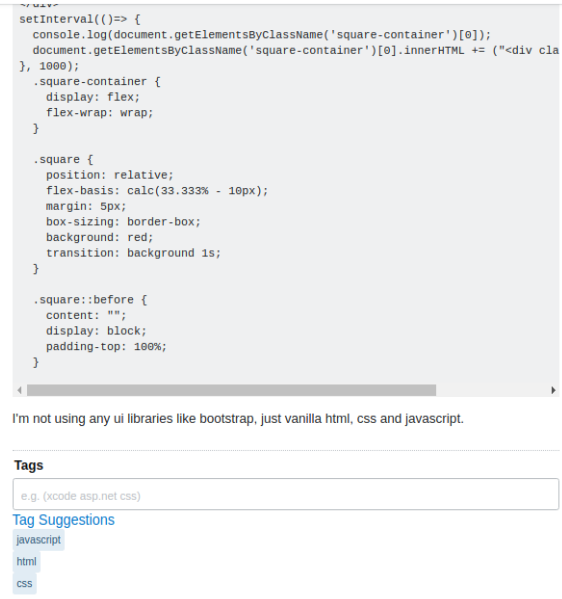

Fig. 6. User Interface of the Classifier.

C. System Usability Scale (SUS)

Using the Google Form containing the System Usability Scale (SUS) questionnaire, the tag recommendation application was evaluated by thirty eight people where 73.7% of them were seniors, 10.5% were juniors, 13.2% were graduates, and 4% were freshmen as shown in the pie graph at Figure 6. All evaluators had a background on computer science because the tag recommendation application was applied to the Stack Overflow website, a question and answer site for programmers.

Figure 8 shows the mean scores of the evaluators for each question. In the SUS, questions 4 and 9 focuses on the

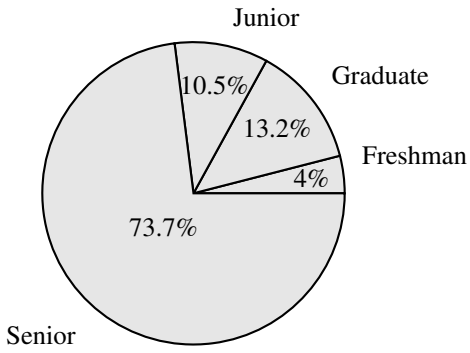

Fig. 7. Classification of SUS evaluators

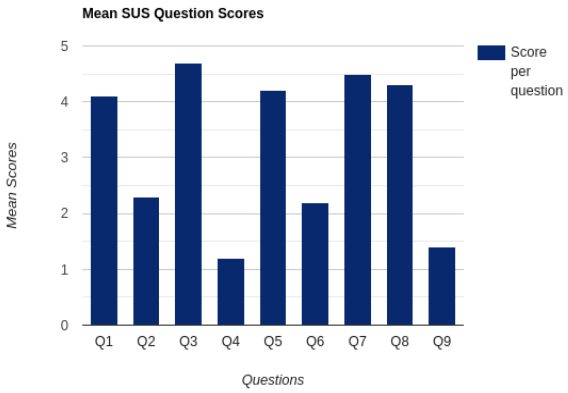

Fig. 8. A bar graph showing the Mean SUS Scores

learnability of the system on a scale from 1 to 5 (from easiest to most difficult learnability). The TagSuggest application achieved a score of 1.2 and 1.4, respectively, which proves that the application is easy to learn on how to use it.

Questions 2 and 6 focuses on the complexity and consistency of the application on a scale from 1 to 5 (1 being the least complex and least inconsistent; 5 being the most complex and most inconsistent). The TagSuggest achieved a score of 2.3 and 2.2, respectively.

The other questions evaluate on the usability of the application (1 being the least usable and 5 being the most usable). On average, the TagSuggest got a score of 4.36 in terms of usability.

Overall, the TagSuggest application achieved a score of 74.04 which is considerably above average [44].

V. Conclusion and Future Work

In this study, a chrome extension application that suggests multi-label tags to Stack Overflow questions was created in order to reduce the burden of manual tagging and improperly tagged questions.

Based on the results, the Support Vector Machine attained the highest accuracy, precision, recall, and f1 score with a score of 94.15486%, 91.05935%, 92.34636%, and 90.84867%,

respectively. SVM was therefore used as a classifier to the TagSuggest chrome extension application.

In the results of the SUS evaluation, the application scored 74.4079 which was considered to be above average. It can then be concluded that the system provides a satisfactory level in terms of usability, consistency, and learnability.

REFERENCES

[1] V. S. Rekha and S. Venkatapathy, "Understanding the usage of online forums as learning platforms," *Procedia Computer Science*, vol. 46, pp. 499–506, 2015.

[2] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[3] S. E. Inc., *Stack Exchange API v2.2*, (accessed October 1, 2018), https://api.stackexchange.com/docs.

[4] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang, "Personalized tag recommendation using graph-based ranking on multi-type interrelated objects," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2009, pp. 540–547.

[5] M. Eric, A. Klimovic, and V. Zhong, "{meric, anakli, vzhong}@ stanford. edu december 8, 2014," 2014.

[6] L. Short, C. Wong, and D. Zeng, "Tag recommendations in stackoverflow," *CS224W Project*, 2014.

[7] S. M. Liu and J.-H. Chen, "A multi-label classification based approach for sentiment classification," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1083–1093, 2015.

[8] T. W. Edgar and D. O. Manz, *Research Methods for Cyber Security*. Syngress, 2017.

[9] C. Keming and Z. Jianguo, "Research on the text classification based on natural language processing and machine learning," *JOURNAL OF THE BALKAN TRIBOLOGICAL ASSOCIATION*, vol. 22, no. 3, pp. 2484–2494, 2016.

[10] S.-L. Developers, *An introduction to machine learning with scikit-learn*, (accessed October 1, 2018), http://scikit-learn.org/stable/tutorial/basic/tutorial.html.

[11] E. Alpaydın, "Introduction to machine learning 2nd edition ed," 2010.

[12] X. Kong, S. Gong, L. Su, N. Howard, and Y. Kong, "Automatic detection of acromegaly from facial photographs using machine learning methods," *EBioMedicine*, vol. 27, pp. 94–102, 2018.

[13] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[14] N. Project, *Natural Language Toolkit*, May 06, 2018 (accessed November 8, 2018), https://www.nltk.org/#natural-language-toolkit.

[15] S. Bird, "Nltk-lite: Efficient scripting for natural language processing," in *Proceedings of the 4th International Conference on Natural Language Processing (ICON)*, 2005, pp. 11–18.

[16] R. Wang, G. Chen, and X. Sui, "Multi label text classification method based on co-occurrence latent semantic vector space," *Procedia computer science*, vol. 131, pp. 756–764, 2018.

[17] A. M. Almeida, R. Cerri, E. C. Paraiso, R. G. Mantovani, and S. B. Junior, "Applying multi-label techniques in emotion identification of short texts," *Neurocomputing*, 2018.

[18] K. Nooney, *Deep dive into multi-label classification! towards data science*, 2018, https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff.

[19] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[20] B. Krawczyk, M. Galar, M. Woźniak, H. Bustince, and F. Herrera, "Dynamic ensemble selection for multi-class classification with one-class classifiers," *Pattern Recognition*, vol. 83, pp. 34–51, 2018.

[21] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[22] R. Gholami and N. Fakhari, "Support vector machine: Principles, parameters, and applications," in *Handbook of Neural Computation*. Elsevier, 2017, pp. 515–535.

[23] Saedsayad, *Logistic Regression*, 2018 (accessed December 1, 2018), https://www.saedsayad.com/logistic_regression.htm.

[24] T. S. Inc, *Support Vector Machines*, 2018 (accessed November 2, 2018), http://www.statsoft.com/textbook/support-vector-machines.

[25] H. Kandan, *Understanding the kernel trick*, August 31, 2017 (accessed November 2, 2018), https://towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78.

[26] N. Shafiabady, L. H. Lee, R. Rajkumar, V. Kallimani, N. A. Akram, and D. Isa, "Using unsupervised clustering approach to train the support vector machine for text classification," *Neurocomputing*, vol. 211, pp. 4–10, 2016.

[27] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[28] C. U. Press, *Naive Bayes text classification*, 2018 (accessed March 23, 2019), https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html.

[29] T. Inc., *Naive Bayes*, 2018 (accessed December 1, 2018), https://www.techopedia.com/definition/32335/naive-bayes.

[30] Z. Wei, H. Zhang, Z. Zhang, W. Li, and D. Miao, "A naive bayesian multi-label classification algorithm with application to visualize text search results," *International Journal of Advanced Intelligence*, vol. 3, no. 2, pp. 173–188, 2011.

[31] A. McCallum, "Multi-label text classification with a mixture model trained by em," in *AAAI workshop on Text Learning*, 1999, pp. 1–7.

[32] M.-L. Zhang, J. M. Peña, and V. Robles, "Feature selection for multi-label naive bayes classification," *Information Sciences*, vol. 179, no. 19, pp. 3218–3229, 2009.

[33] S. Solution, *What is Logistic Regression?*, 2018 (accessed December 1, 2018), https://www.statisticssolutions.com/what-is-logistic-regression/.

[34] J. Brownlee, "A gentle introduction to the bag-of-words model," *Machine Learning Mastery*, vol. 21, 2017.

[35] A. Fujino and H. Isozaki, "Multi-label classification using logistic regression models for ntcir-7 patent mining task." in *NTCIR*, 2008.

[36] K. Hewa, *An introduction to Grid Search*, 2019 (accessed April 13, 2019), https://medium.com/datadriveninvestor/an-introduction-to-grid-search-ff57adcc0998.

[37] E. Lutins, *Grid Searching in Machine Learning: Quick Explanation and Python Implementation*, 2019 (accessed April 13, 2019), https://medium.com/@elutins/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-55(

[38] Y. Wu, S. Xi, Y. Yao, F. Xu, H. Tong, and J. Lu, "Guiding supervised topic modeling for content based tag recommendation," *Neurocomputing*, vol. 314, pp. 479–489, 2018.

[39] R. Krestel and P. Fankhauser, "Personalized topic-based tag recommendation," *Neurocomputing*, vol. 76, no. 1, pp. 61–70, 2012.

[40] S. Schuster, W. Zhu, and Y. Cheng, "Predicting tags for stackoverflow questions," *CS229 Projects, Stanford university*, 2013.

[41] U. Erra, S. Senatore, F. Minnella, and G. Caggianese, "Approximate tf–idf based on topic extraction from massive message stream using the gpu," *Information Sciences*, vol. 292, pp. 143–161, 2015.

[42] P. Lingras and C. J. Butz, "Precision and recall in rough support vector machines," in *Granular Computing, 2007. GRC 2007. IEEE International Conference on*. IEEE, 2007, pp. 654–654.

[43] J. Sauro, *Measuring Usability with The System Usability Scale*, 2016 (accessed March 23, 2019), https://www.userfocus.co.uk/articles/measuring-usability-with-the-SUS.html.

[44] usability.gov, *System Usability Scale*, 2019 (accessed April 7, 2019), https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

Trixia R. Belleza is a BS Computer Science student. She enjoys seeking knowledge about various algorithms that are in-line with her chosen degree.