International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

Tagging Stack-Overflow Questions using Supervised Machine Learning Techniques

Ms. Garima Gupta
Asst. Prof.,
MAIT, Delhi

Harshit Aggarwal
Student,
MAIT, Delhi

Disha Sharma
Student,
MAIT, Delhi

Ishan Agarwal
Student,
MAIT, Delhi

***Abstract*:- Lately, there has been a tremendous improvement in in- novation and the rise of new ones. This is joined by an ex- pansion in inquiries on sites, for example, Stack-Overflow. These inquiries should be ordered in view of labels. The issue of label proposition arises when the client gives no labels, or the client showed labels are inconsequential to the request. Our venture plans to determine this issue via auto-labeling questions utilizing directed AI procedures.**

Keywords— Natural Language Processing, Information Retrieval, Supervised Learning, SVM, Random Forest , SGD Classi-fier, MultinomialNB, LinearSVC, LogisticRegression

1. INTRODUCTION

Tagging gives a helpful means to benefit out distinguishing proof tokens to explore papers, articles, or questions, which work with the recommendation, search, and attitude cycle of inquiries. It gives a way to deal with sharing and partner applicable watch- words or tags. The issue of label recommendation emerges when the client gives no tags, or client indicated tags are insignificant to the archive. These sorts of client focused approaches are not ex- ceptionally functional for label recommendation. In tagging sys- tems, manual tagging becomes blunder inclined and tedious when there are an excessive number of papers or questions. This venture will move toward the auto-tagging of StackOverflow questions uti- lizing supervised machine learning techniques and will return pre- cise tags connected with the inquiry. One of the critical pieces of the review is separating and pre-processing the dataset to extricate the most pertinent tags and catchphrases. The methodology de- pends on removing catchphrases utilizing the TF-IDF vectorizer weighted score [1] and afterward applying supervised machine learning calculations to prepare the machine learning model.

We explored different datasets such arxiv library dataset and the 10% StackOverflow QA dataset [2]. We chose the StackOver- flow database as it matches our exploration of other research pa- pers, and we can more efficiently apply supervised machine learn- ing algorithms to this dataset. The dataset contains four relevant columns: the title, question, answer, and tags. We consolidate this data and remove irrelevant columns and tags. Then we fur- ther classify tags based on the most frequently seen tags. We then vectorize the data and title and consolidate it in a single column. Further, we apply necessary pre-processing for various algorithms, namely, Multinomial Naive Bayes Algorithm, Logistic Regression Algorithm [3], Support Vector Machine (SVM) Algorithm [4], and Stochastic Gradient Descent (SGD) Classifier. Further, we com- pare the algorithms' results and represent outcomes by comparing them on parameters such as Hamming score and Precision.

2. LITERATURE REVIEW

Paper 1:

In the paper titled Supervised ML-based approach for auto-tagging of scientific literature [5],The paper explores supervised Machine Learning approaches to provide tags for documents related to scientific domain. They approach the solution using natural Language Processing(NLP) by extracting keywords and applying appropriate text classification Techniques.in this paper they have taken arxiv dataset, which is a dataset of library of Cornell university which have data of more than millions of documents.This dataset is based on the ACM Computing Classification System. In this paper main feature extraction techinques that is used includes word embeddings and text vectorization. The paper concludes by giving best result using tf-idf vectorization and using Support Vector Classifier as the algorithm for training.

Paper 2:

In the paper named Natural Language Processing for Information Extraction [6] This examination paper is about Natural Lan- guage Processing (NLP) for information extraction. With the as- cent of the digital age, there is a blast of information in news, arti- cles, web-based entertainment, etc. Quite a bit of this information lies in nebulous structure, and physically overseeing and really uti-lizing it is drawn-out, dull, and work concentrated. This blast of information and the requirement for more refined and proficient in- formation dealing with apparatuses brings about Information Ex- traction (IE) and Information Retrieval (IR) innovation. Informa- tion Extraction systems accept natural language text as informa- tion and produce organized information indicated by unambigu- ous rules pertinent to a specific application. Different sub-errands of IE, for example, Named Entity Recognition, Coreference Res- olution, Named Entity Linking, Relation Extraction, and Knowl- edge Base thinking structure the structure

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

blocks of different very good quality Natural Language Processing (NLP) assignments, for example, Machine Translation, Question-Answering System, Natural Language Understanding, Text Summarization and Dig- ital Assistants like Siri, Cortana and Google Now. This paper presents Information Extraction innovation, and its different sub- undertakings feature best in class research in various IE subtasks, momentum difficulties, and future examination headings.

Paper 3:

In the paper named Automatically Labeling Low Quality Con- tent on Wikipedia By Leveraging Patterns in Editing Behaviors [7] creators start by spreading out A methodology in light of meta- physics for auto-tagging is proposed in.Ontology is an informa- tion model in which terms are addressed as an ordered progres- sion. It incorporates characterization and tag-choice. Term-weight matrix and cosine-similarity is utilized for the characterization cy- cle. Label choice interaction relies upon metaphysics weight. A contribution of a huge train dataset comprising of title, unique is utilized and TF-IDF is utilized for building term weight matrix. In this work, labels are positioned in terms of recurrence as well as in terms of similarity moreover. A cross breed approach is uti- lized for removing the header information which can be utilized for investigating the elements in the examination region among research networks. Data integration and validation utilizing extri- cated header information assets like GROBID, Parsit are utilized as it is contended that any a single instrument can't give effectiveoutcomes on all example research articles.

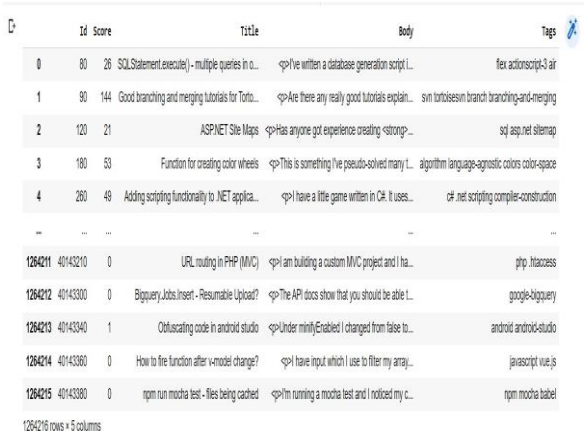
3. METHODOLOGY

• We choose the 10% StackOverflow questions dataset and use the Python Pandas library to import the dataset from the CSV file. Then we create two files, i.e., the questions.csv file and the Tags.csv file, in which we get the title body and tags related to the question.

	Id	Tag
0	80	flex
1	80	actionscript-3
2	80	air
3	90	svn
4	90	tortoisesvn
...	...	...
3750989	40143360	javascript
3750990	40143360	vue.js
3750991	40143380	npm
3750992	40143380	mocha
3750993	40143380	babel

3750994 rows × 2 columns

• The first step of pre-processing involves consol- idating the questions and tags into single data from the input dataset and mapping the labels ac- cording to the question id given in the dataset.



	Id	Score	Title	Body	Tags
0	80	26	SQLStatement.execute() - multiple queries in o...	<p>I've written a database generation script i...	flex actionscript-3 air
1	90	144	Good branching and merging tutorials for Torto...	<p>Are there any really good tutorials explain...	svn tortoisesvn branch branching-and-merging
2	120	21	ASP.NET Site Maps	<p>Has anyone got experience creating ...	sql asp.net sitemap
3	180	53	Function for creating color wheels	<p>This is something I've pseudo-solved many t...	algorithm language-agnostic colors color-space
4	260	49	Adding scripting functionality to .NET applica...	<p>I have a little game written in C#. It uses...	c# .net scripting compiler-construction
...	...	...	...	...	...
1264211	40143210	0	URL routing in PHP (MVC)	<p>I am building a custom MVC project and I ha...	php .htaccess
1264212	40143300	0	Bigquery.Jobs.Insert - Resumable Upload?	<p>The API docs show that you should be able t...	google-bigquery
1264213	40143340	1	Obfuscating code in android studio	<p>Under minifyEnabled I changed from false to...	android android-studio
1264214	40143360	0	How to fire function after v-model change?	<p>I have input which I use to filter my array...	javascript vue.js
1264215	40143380	0	npm run mocha test - files being cached	<p>I'm running a mocha test and I noticed my c...	npm mocha babel

1264216 rows × 5 columns

• The next step involved the grouping of tags and removal of repeated tags.

• Then we filtered the data in the questions data frame, and we dropped the columns such as 'ownerUserId', 'CreationDate', 'ClosedDate'.

```
[11] 1 question_df.drop(columns=['OwnerUserId', 'CreationDate', 'ClosedDate'], inplace=True)
     2 question_df = question_df.merge(grouped_tags_final, on='Id')
     3 question_df
```

• Then we further disregarded the rows having scores less than 5. The score is the sum of up- votes and negative downvotes for a question.

```
1 #now we filter as per the scores
2 print(f"Minimum Score: {question_df['Score'].min()}")
3 print(f"Maximum Score: {question_df['Score'].max()}")
4 #deleting queries with score less than 5
5 new_question_df = question_df[question_df['Score'] > 5]
```

```
Minimum Score: -73
Maximum Score: 5190
```

• After filtering based on id and score, we dropped these columns because they are not needed for model training.

```
[14] 1 new_question_df.drop(columns=['Id', 'Score'], inplace=True)
     2 new_question_df
```

	Title	Body	Tags
0	SQLStatement.execute() - multiple queries in o...	<p>I've written a database generation script i...	flex actionscript-3 air
1	Good branching and merging tutorials for Torto...	<p>Are there any really good tutorials explain...	svn tortoisesvn branch branching-and-merging

• Then we processed the tags and filtered unique tags; we ran frequency distribution using NLTK and found that more than 14000+ tags occurred more than 220000+ times. So, using this data, we extracted the top 100 tags in the dataset, which came out to be primarily related to the tech stack.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

```
[15] 1 new_question_df['Tags'] = new_question_df['Tags'].apply(lambda x: x.split())
     2 unique_tags = list(set([item for sublist in new_question_df['Tags'].values for item in sublist]))
     3 len(unique_tags)
```

```
14883
```

```
[16] 1 flat_list = [item for sublist in new_question_df['Tags'].values for item in sublist]
     2 keywords = nltk.FreqDist(flat_list)
     3 keywords = nltk.FreqDist(keywords)
     4 sum(keywords.values())
```

```
224129
```

```
1 flat_list = [item for sublist in new_question_df['Tags'].values for item in sublist]
2 keywords = nltk.FreqDist(flat_list)
3 frequencies_words = keywords.most_common(100)
4 tags_features = [word[0] for word in frequencies_words]
5
6 print(len(tags_features))
7
8 tags_features
```

```
100
['c#',
 'java',
 'javascript',
 'android',
 'python',
 'c++',
 'php',
 'jquery',
 '.net',
 'ios',
 'html',
 'css',
 'c',
 'iphone',
 'objective-c',
 'ruby-on-rails',
 'sql',
 'asp.net',
 'mysql',
 'ruby'.
```

• The next task was to filter the content, body, and title column data. We need to remove HTML tags and unnecessary elements from the text,and we used the bs4 library for this purpose.

```
1 # Converting html to text in the body
2 new_question_df['Body'] = new_question_df['Body'].apply(lambda x: BeautifulSoup(x).get_text())
3 new_question_df
```

	Title	Body	Tags
1	good branch merge tutorials tortoisesvn	really good tutorials explain branch merge apa...	[svn]
2	asp.net site map	anyone get experience create sql-based asp.net...	[sql, asp.net]
3	function create color wheel	something pseudo-solved many time never quite ...	[algorithm]
4	add script functionality .net applications	little game write c use database back-end trad...	[c#, .net]
5	use nest class case	work collection class use video playback recor...	[c++, oop, class]

• Then we do the essential part of the project, which is pro- cessing the 'body' column. Here, we first clean the data, i.e., remove the redundant spaces between the words, and then we clean the punctuation in which we removed the special characters, if any, present in the data.

• Then we perform lemmatization on the data. Lemmatization refers to reduce the word to simplest form , such that its vo-cabulary and morphological analysis

• Then we applied StopWords removal using Natural Language Toolkit, where the stopwords in NLTK refers to word in documents which occur frequently. Also stopwords are the word which do not define the con-text to the document so it is essential to remove them.

```
1 def clean_text(text):
2     text = text.lower()
3     text = text.strip(' ')
4     return text
5
6 token = ToktokTokenizer()
7 punct = '!"#$%&\'()*+,./:;<=>?@[\\]^_`{|}~'
8 lemma = WordNetLemmatizer()
9 stop_words = set(stopwords.words("english"))
10
11 def strip_list_noempty(mylist):
12     newlist = (item.strip() if hasattr(item, 'strip') else item for item in mylist)
13     return [item for item in newlist if item != '']
14
15 def clean_punct(text):
16     words = token.tokenize(text)
17     punctuation_filtered = []
18     regex = re.compile('[%s]' % re.escape(punct))
19     remove_punctuation = str.maketrans(' ', ' ', punct)
20     for w in words:
21         if w in tags_features:
22             punctuation_filtered.append(w)
23         else:
24             punctuation_filtered.append(regex.sub('', w))
25     filtered_list = strip_list_noempty(punctuation_filtered)
26     return ' '.join(map(str, filtered_list))
27
28 def lemmatizeWords(text):
29     words = token.tokenize(text)
30     listLemma = []
31     for w in words:
32         x = lemma.lemmatize(w, pos="v")
33         listLemma.append(x)
34     return ' '.join(map(str, listLemma))
35
36 def stopWordsRemove(text):
37     stop_words = set(stopwords.words("english"))
38     words = token.tokenize(text)
39     filtered = [w for w in words if not w in stop_words]
40     return ' '.join(map(str, filtered))
```

```
[23] 1 # Remove stopwords, punctuation and lemmatize for text in body
     2 new_question_df['Body'] = new_question_df['Body'].apply(lambda x: clean_text(x))
     3 new_question_df['Body'] = new_question_df['Body'].apply(lambda x: clean_punct(x))
     4 new_question_df['Body'] = new_question_df['Body'].apply(lambda x: lemmatizeWords(x))
     5 new_question_df['Body'] = new_question_df['Body'].apply(lambda x: stopWordsRemove(x))
     6
     7 # Remove stopwords, punctuation and lemmatize for title.
     8 new_question_df['Title'] = new_question_df['Title'].apply(lambda x: str(x))
     9 new_question_df['Title'] = new_question_df['Title'].apply(lambda x: clean_text(x))
    10 new_question_df['Title'] = new_question_df['Title'].apply(lambda x: clean_punct(x))
    11 new_question_df['Title'] = new_question_df['Title'].apply(lambda x: lemmatizeWords(x))
    12 new_question_df['Title'] = new_question_df['Title'].apply(lambda x: stopWordsRemove(x))
```

```
[24] 1 #data is finally cleaned and preprocessed
     2 new_question_df
```

	Title	Body	Tags
1	good branch merge tutorials tortoisesvn	really good tutorials explain branch merge apa...	[svn]
2	asp.net site map	anyone get experience create sql-based asp.net...	[sql, asp.net]
3	function create color wheel	something pseudo-solved many time never quite ...	[algorithm]
4	add script functionality .net applications	little game write c use database back-end trad...	[c#, .net]
5	use nest class case	work collection class use video playback recor...	[c++, oop, class]
...	...	...	...
1262834	stl list bad performance	suppose pushback popfront methods stl list imp...	[c++, list]
1262915	use dict subset dataframe	say give dataframe columns categorical data da...	[python]
1263065	way use itertools python clean nest iterations	let say follow code 123 b 246 c 357 j b k c pr...	[python]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

- Then we used Mulilabel binarizer to convert the tags into binary data. The step is also called Bucketization.

```
[25]  1 X1 = new_question_df['Body']
      2 X2 = new_question_df['Title']
      3 y = new_question_df['Tags']
      4
      5 multilabel_binarizer = MultiLabelBinarizer()
      6 y_bin = multilabel_binarizer.fit_transform(y)
      7 y_bin
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

- Then we used the TF-IDF vectorizer to obtain scores of different words in the documents, and then we applied the vectorization to the dataset.

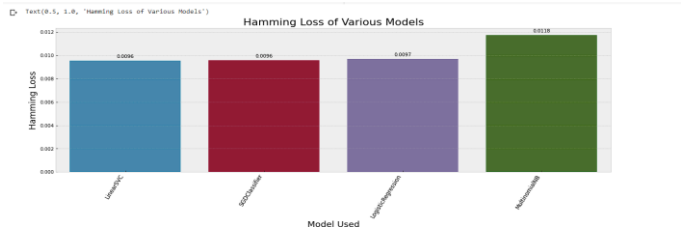
```
1 vectorizer_X1 = TfidfVectorizer(analyzer = 'word',
2                                 min_df=0.0,
3                                 max_df = 1.0,
4                                 strip_accents = None,
5                                 encoding = 'utf-8',
6                                 preprocessor=None,
7                                 token_pattern=r"(?u)\S\S+",
8                                 max_features=1000)
9
10 vectorizer_X2 = TfidfVectorizer(analyzer = 'word',
11                                 min_df=0.0,
12                                 max_df = 1.0,
13                                 strip_accents = None,
14                                 encoding = 'utf-8',
15                                 preprocessor=None,
16                                 token_pattern=r"(?u)\S\S+",
17                                 max_features=1000)
18
19 X1_tfidf = vectorizer_X1.fit_transform(X1)
20 X2_tfidf = vectorizer_X2.fit_transform(X2)
21
22 X_tfidf = hstack([X1_tfidf,X2_tfidf])
23 X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y_bin, test_size = 0.2, random_state = 0)
24
25 print(X_train)
```

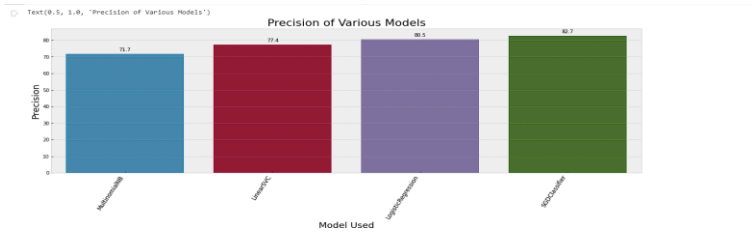
```
  (0, 3)        0.4161645844276208
  (0, 86)       0.46193827545848687
  (0, 110)      0.39314534058953204
  (0, 515)      0.44465220224309476
  (0, 522)      0.44390478800451894
  (0, 761)      0.25315956970731845
  (0, 1001)     0.45713885982738367
  (0, 1052)     0.5926728680752641
  (0, 1526)     0.5067923041014708
  (0, 1777)     0.427696732260936
  (1, 10)       0.14231854988768894
  (1, 41)       0.05733791191004865
  (1, 77)       0.05571315062005313
  (1, 145)      0.07400219715174045
  (1, 174)      0.058858462638403636
  (1, 181)      0.06636976838907754
```

- After processing the dataset, we train the model on 80% of the dataset and then test it on the remaining 20%. Then, we compare the results given by different models based on hamming loss and Precision.

4. RESULTS AND ANALYSIS

After training and testing the model on four different algorithms separately, we compared them based on hamming loss and Precision. The result is shown in the following figures:





The best model is the SGDClassifier model because it has the lowest hamming loss (0.0096) and highest Precision (82.7%). Based on hamming loss alone, the LinearSVC and SGDClassifier model gives the best result with the lowest score of 0.0096.

```
Classifier:  SGDClassifier
Hamming Loss:  0.009573373436757954
Precision:  0.8259963120377815
Recall:  0.45016870624968525
```

```
Classifier:  LogisticRegression
Hamming Loss:  0.009717429159411112
Precision:  0.8040235356334973
Recall:  0.4724278591932316
```

```
Classifier:  MultinomialNB
Hamming Loss:  0.011764286845021371
Precision:  0.7166484546737002
Recall:  0.4353124842624767
```

```
Classifier:  LinearSVC
Hamming Loss:  0.00954567041317081
Precision:  0.7737568537070041
Recall:  0.5249030568565242
```

The **SGDClassifier model** gives the best result with thehighest Precision of **82.6%** based on Precision alone.

5. CONCLUSION

- our solution provides an efficient as well as effecion way of tagging stackoverflow question using relevant tags.
- The imported dataset is processed to perform tagging, and noisy data is removed.
- Our project also generates reports showing the influence of different models and their behavior.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

- The max Precision of the ML model came to be around 80% which suggest we can use it in genuine life software to au- tomate tagging of question to help readers reach the most relevant search results.
- The algorithm can also be improved with time with an in- creasing database of websites like stack overflow, and algo- rithms can also be enhanced with newer deep learning tech-niques.
- This project can be used to recommend and analyze ques- tions based on a specific tag.

REFERENCES

[1] "sklearn.feature extraction.text.TfidfVectorizer." [Online]. Available: – https://scikit-learn/stable/modules/generated/sklearn.feature extraction.text.TfidfVectorizer.html 1

[2] "StackSample: 10% of Stack Overflow Q&A." [On- line]. Available: https://www.kaggle.com/stackoverflow/ stacksample 1

[3] "Logistic Regression in Machine Learning - Javat- point." [Online]. Available: https://www.javatpoint.com/ logistic-regression-in-machine-learning 1

[4] "Support Vector Machine (SVM) Algorithm - Javat- point." [Online]. Available: https://www.javatpoint.com/ machine-learning-support-vector-machine-algorithm 1

[5] M. Zdravković, "Supervised ml-based approach for auto-tagging of scientific literature," in *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2021, pp. 1–5. 1

[6] S. Singh, "Natural language processing for information ex-traction," *arXiv preprint arXiv:1807.02383*, 2018. 2

[7] S. Asthana, S. Tobar Thommel, A. L. Halfaker, and

[8] N. Banovic, "Automatically labeling low quality content on wikipedia by leveraging patterns in editing behaviors," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, no. CSCW2, oct 2021. [Online]. Available: https://doi.org/10.1145/3479503 2