KNOWLEDGE INSTITUTE OF TECHNOLOGY

AUTONOMOUS TAGGING OF STACK OVERFLOW QUESTIONS

APPLIED DATA SCIENCE
(NM2023TMID16853)

MICROSOFT INTERNSHIP PROJECT REPORT

Submitted by

DAMODHARAN M (6112201060301)
SATHYAVEL A S (611220106068)
GOKUL R (611220106022)
PRAVEEN KUMAR R(6112201060318)

ABSTRACT

The autonomous tagging of Stack Overflow questions is an automated process that aims to assign relevant tags to questions posted on the Stack Overflow platform. With the rapid growth of user-generated content, effective tagging plays a crucial role in organizing and categorizing questions, aiding in better searchability and improved user experience. Traditional manual tagging by users is time-consuming and often results in inconsistent or incomplete tags.

To address these challenges, autonomous tagging systems utilize machine learning techniques to automatically assign tags to Stack Overflow questions. This abstract highlights the key aspects of autonomous tagging, including the underlying methods, data sources, and potential benefits.

The process begins by retrieving questions from the Stack Overflow platform through an API. These questions are then processed and prepared for feature extraction, which involves transforming the textual content into meaningful representations. Various features such as the question's title, body, and related metadata are extracted to capture the essential characteristics.

The benefits of autonomous tagging are manifold. It reduces the burden on users to manually tag questions, resulting in consistent and accurate tags. It enhances the searchability of questions by associating them with relevant tags, improving the accuracy of search results. Additionally, autonomous tagging can facilitate better content recommendation, allowing users to discover similar questions or related topics of interest.

In conclusion, autonomous tagging of Stack Overflow questions leverages machine learning techniques to automate the process of assigning relevant tags. This approach improves the organization and searchability of questions,

ultimately enhancing the user experience and knowledge sharing within the Stack Overflow community.

INTRODUCTION

1.1 ProjectOverview

OBJECTIVE:

1.Improve tagging accuracy: The project's primary objective is to enhance the accuracy of question tagging on Stack Overflow by reducing the reliance on manual tagging. By automating this process, the system aims to assign relevant tags more consistently and effectively.

2.Increase efficiency: The autonomous tagging system intends to improve the efficiency of tagging questions by reducing the time and effort required from human moderators or question askers. This allows them to focus on other important tasks while still ensuring accurate tagging.

3.Enhance user experience: By providing more accurate and relevant tags to questions, the project aims to enhance the overall user experience on Stack Overflow. Users will benefit from improved search functionality, better question categorization, and higher-quality recommended.

APPROACH:

Data collection: The project will begin by gathering a large dataset of Stack Overflow questions, including their associated tags. This dataset will serve as the foundation for training and evaluating the autonomous tagging system.

Model development: Various machine learning and NLP techniques will be explored to build an effective tagging model. This could involve approaches

such as deep learning architectures (e.g., recurrent neural networks or transformer models) or traditional supervised learning algorithms.

Training and evaluation: The model will be trained on the collected dataset using appropriate techniques, and its performance will be evaluated using metrics like precision, recall, and F1-score. The model will be iteratively refined to achieve higher accuracy.

1.2 Proposed solution

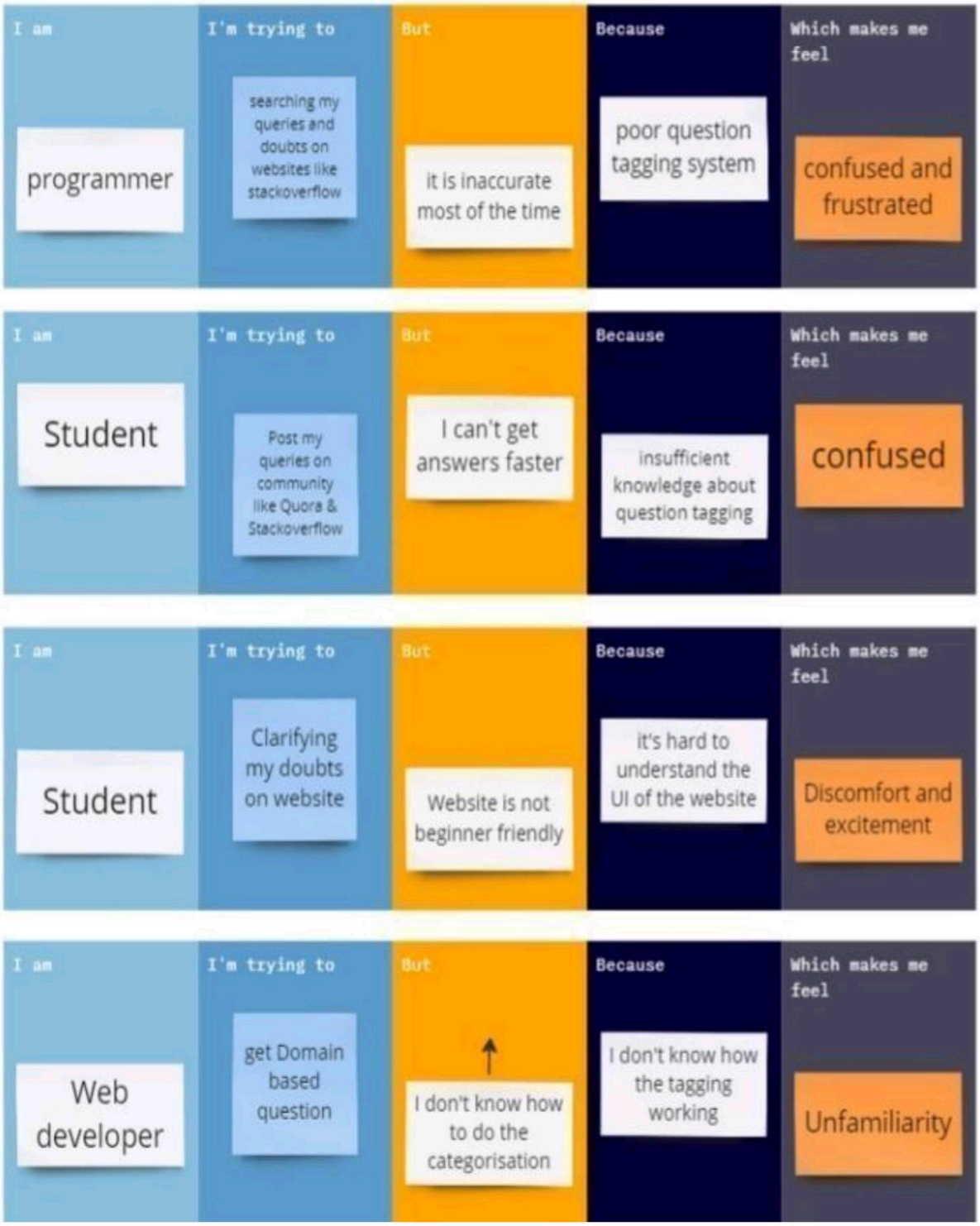
Manual tagging of questions on Stack Overflow can be subjective and inconsistent. The autonomous tagging system aims to enhance the accuracy of tag assignment by leveraging machine learning algorithms and NLP techniques. By analyzing the content of the questions and considering various factors, the system can assign relevant and appropriate tags more consistently. Accurate tagging plays a crucial role in enhancing the user experience on Stack Overflow. It enables users to find relevant questions more easily using search functionality and filters based on tags. By automating the tagging process, the project aims to improve question categorization and increase the overall satisfaction of users seeking answers and information on the platform. Tags play a significant role in the search functionality of Stack Overflow. By automating the tagging process and ensuring accurate and relevant tags are assigned, the project aims to improve search results and recommendations. Users can expect more precise and targeted search outcomes, leading to faster and more effective problem-solving.

2. IDEATION & PROPOSED SOLUTION

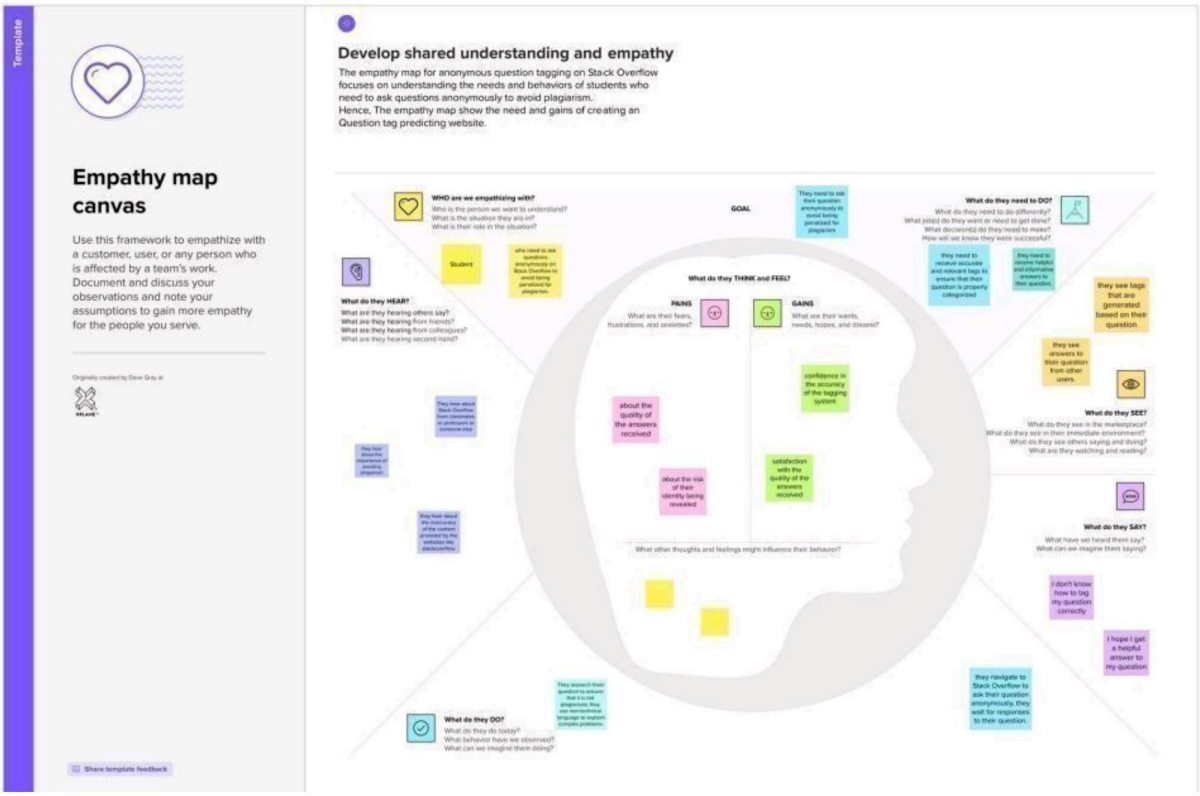
1. Problem statement definition

Users face difficulties in finding relevant questions due to inconsistencies in manually assigned tags. Different human moderators or question askers may apply varying tags to similar questions, leading to confusion and reduced search effectiveness. Manual tagging requires significant time and effort from human moderators or question askers. They need to analyze the question content, consider the appropriate tags, and assign them accurately. This

process is laborious and can slow down the overall question management workflow. Some question askers may not possess sufficient expertise or knowledge to assign appropriate tags to their questions. As a result, the assigned tags may be inaccurate or inadequate, leading to reduced visibility and limited responses from the community.

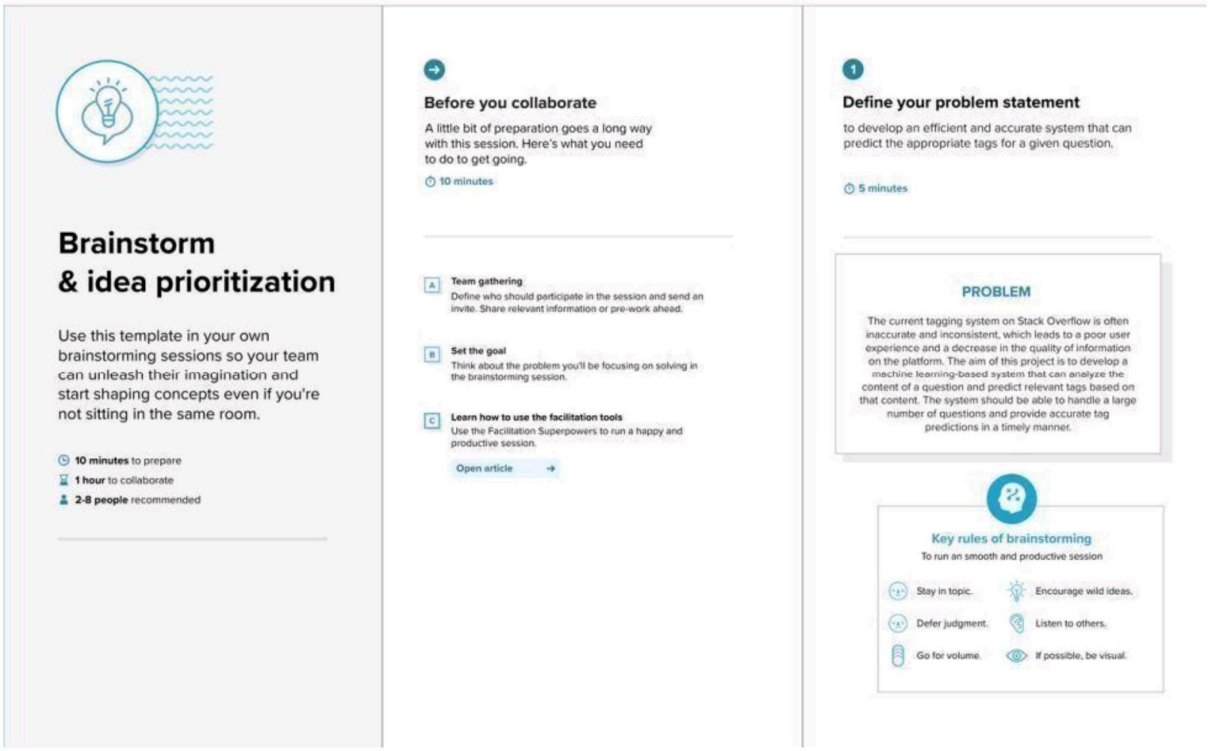

I am
I'm trying to
But
Because
Which makes me feel
programmer
searching my queries and doubts on websites like stackoverflow
it is inaccurate most of the time
poor question tagging system
confused and frustrated
I am
I'm trying to
But
Because
Which makes me feel
Student
Post my queries on community like Quora & Stackoverflow
I can't get answers faster
insufficient knowledge about question tagging
confused
I am
I'm trying to
But
Because
Which makes me feel
Student
Clarifying my doubts on website
Website is not beginner friendly
it's hard to understand the UI of the website
Discomfort and excitement
I am
I'm trying to
But
Because
Which makes me feel
Web developer
get Domain based question
I don't know how to do the categorisation
I don't know how the tagging working
Unfamiliarity

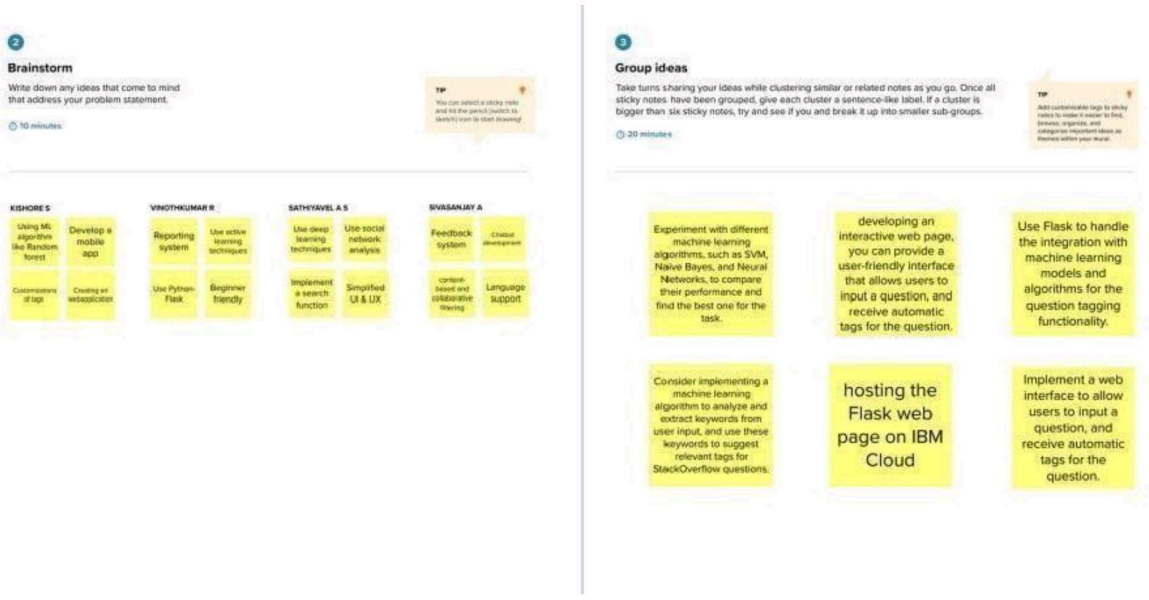
2.2 Empathy Map Canvas: The Empathy Map Canvas helps understand the perspectives, needs, and challenges of Stack Overflow users regarding the tagging process. It allows for a deeper understanding of their emotions, thoughts, and behaviors , which can guide the development of an autonomous tagging system that addresses their pain points and provides valuable gains.

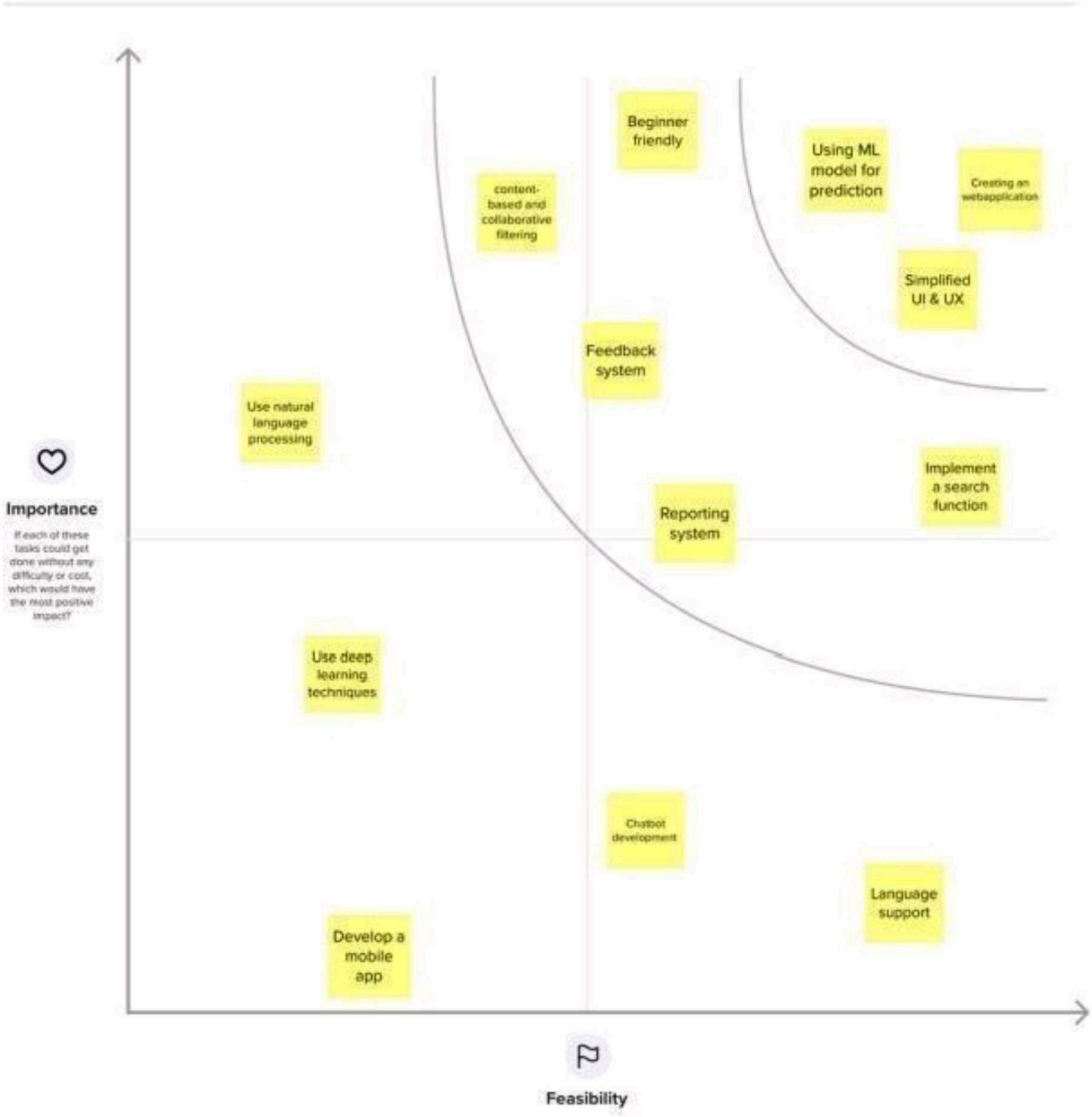


2.3 Ideation & Brainstorming

Explore various machine learning models such as recurrent neural networks (RNS) or transformer models for tag prediction .Consider using pre-trained language models like BERT or GPT to capture the context and semantics of the questions. Investigate ensemble models that combine the strengths of multiple algorithms to improve tagging accuracy. Develop an NLP model that can analyze the question content and suggest appropriate tags based on their context. The model could be trained on a large dataset of tagged questions and validated using a testing dataset to ensure accuracy.


Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
10 minutes to prepare
1 hour to collaborate
2-8 people recommended
Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes
A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.
C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article
1
Define your problem statement
to develop an efficient and accurate system that can predict the appropriate tags for a given question.
5 minutes
PROBLEM
The current tagging system on Stack Overflow is often inaccurate and inconsistent, which leads to a poor user experience and a decrease in the quality of information on the platform. The aim of this project is to develop a machine learning-based system that can analyze the content of a question and predict relevant tags based on that content. The system should be able to handle a large number of questions and provide accurate tag predictions in a timely manner.
Key rules of brainstorming
To run an smooth and productive session
Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.


2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
KISHORE S
Using ML algorithm like Random forest
Develop a mobile app
Customizations of tags
Creating an webapplication
VINOTHKUMAR R
Reporting system
Use active learning techniques
Use Python-Flask
Beginner friendly
SATHYAVEL A S
Use deep learning techniques
Use social network analysis
Implement a search function
Simplified UI & UX
SIVASANJAY A
Feedback system
Chatbot development
content-based and collaborative filtering
Language support
3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
Experiment with different machine learning algorithms, such as SVM, Naive Bayes, and Neural Networks, to compare their performance and find the best one for the task.
developing an interactive web page, you can provide a user-friendly interface that allows users to input a question, and receive automatic tags for the question.
Use Flask to handle the integration with machine learning models and algorithms for the question tagging functionality.
Consider implementing a machine learning algorithm to analyze and extract keywords from user input, and use these keywords to suggest relevant tags for StackOverflow questions.
hosting the Flask web page on IBM Cloud
Implement a web interface to allow users to input a question, and receive automatic tags for the question.



2.4 Proposed Solution

ML algorithms can be trained on a large dataset of Stack Overflow questions and their corresponding tags. Features like the question title, body, code snippets, and existing tags can be used to train a model to predict appropriate tags for new questions. Various ML algorithms such as classification or regression models can be explored, and techniques like multi-label classification can be used to assign multiple

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	When users ask a question on Stack Overflow, they are required to add tags to their post to help categorize it and make it easier for others to find. However, users may not always choose the most appropriate tags, either because they are not familiar with the relevant terminology or because they are in a hurry and do not take the time to carefully consider the most relevant tags.
2.	Idea / Solution description	Creating a website : This would involve creating a layout, selecting colors and fonts, and determining the overall look and feel of the site. To develop the website using the chosen programming languages and frameworks. The website would include a user-friendly interface that allows users to easily submit their questions and receive suggested tags. Users would also be able to manually edit or add tags as needed.
3.	Novelty / Uniqueness	User Feedback Loop: Allow users to provide feedback on the suggested tags to improve the accuracy of the machine learning model and This feedback can be used to retrain the model on the new data

4.	Social Impact / Customer Satisfaction	Improving accessibility to knowledge: Stack Overflow is a popular online community of developers where people can ask and answer technical questions. Building a stronger community: By creating a more efficient and effective tagging process, Stack Overflow can improve the quality of content and interactions within the community.
5.	Business Model (Revenue Model)	Enhancing search functionality: Improved question tagging can help to enhance the search functionality of the platform, making it easier for users to find the answers they are looking for. Reducing moderation costs: By reducing the number of incorrectly tagged questions, the platform can reduce the burden on moderators who are responsible for ensuring the quality of content on the site.

6.	Scalability of the Solution	Cloud hosting: Hosting the website on a cloud hosting service, Amazon Web Services (AWS) or Microsoft Azure, can allow for easy scaling of resources, CPU, RAM, and storage, number of users and the amount of data being processed increases. Monitoring and optimization: Regular monitoring of website performance and optimization of system resources can help ensure that the website can handle increasing loads over time.

3. Requirement Analysis

1. Functional Requirements

Following are the functional requirements of the proposed solution.

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Data Collection	Gather a large dataset of Stack Overflow posts specifically related to functional requirements. This dataset should include a diverse range of posts covering various domains and technologies
FR-2	Preprocessing	Clean and preprocess the collected data to remove irrelevant information, such as code snippets or irrelevant comments, and perform text normalization techniques like stemming or lemmatization. This step helps in standardizing the text data and reducing noise.

FR-3	Feature Extraction	Extract relevant features from the preprocessed text to represent the posts in a numerical or vector format. This can be done using techniques like bag-ofwords, TF-IDF , or word embeddings to capture the semantic meaning of the text.
FR-4	Training Data Preparation	Annotate a subset of the collected data with appropriate functional requirement tags or labels. This involves manually tagging the posts based on their functional characteristics. The annotated data will serve as the training set for the machine learning model.
FR-5	Model Training	Utilize machine learning algorithms to train a model on the annotated data. Common approaches include supervised learning algorithms like Naive Bayes, Support Vector Machines (SVM), or neural network architectures such as recurrent neural networks (RNNs) or transformers.
FR-6	Model Evaluation and Iteration	Evaluate the trained model's performance using appropriate evaluation metrics, such as precision, recall, and F1 score. Fine-tune the model and iterate on the training process by incorporating user feedback and retraining the model as needed.
FR-7	Flask Integration	The created model should be integrated with flask to provide an web based interface to the user.

3.2 Non-Functional Requirements

Following are the non-functional requirements of the proposed solution.

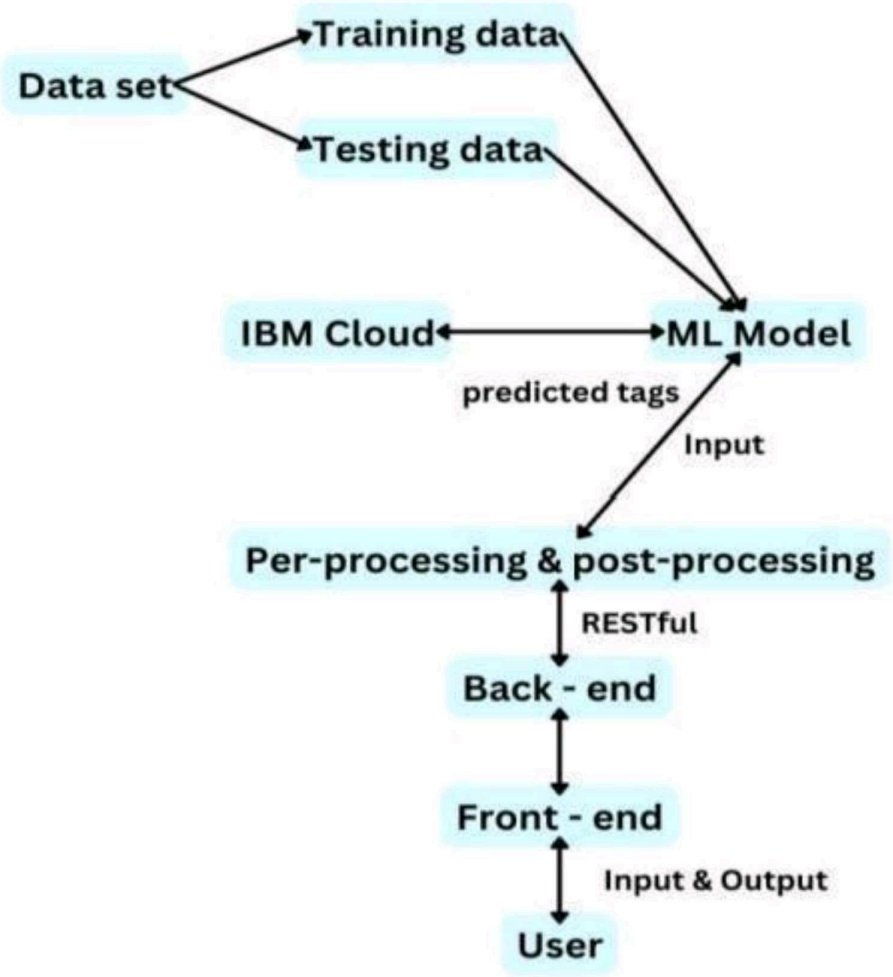
FR No	Non-Functional Requirement	Description
NFR-1	Adaptability	The system should be adaptable to changes in the Stack Overflow platform, community preferences, and evolving programming languages or technologies. It should be able to learn from feedback and adapt its tagging algorithms to improve over time.
NFR-2	**Security**	It should handle user information and question data in a secure manner and only use it for the purpose of tagging questions
NFR-3	**Reliability**	The system will be responsible for detecting and tracking vehicles in real time, it is crucial that it operates reliably and consistently without any significant downtime or failure
NFR-4	**Performance**	The system should be able to process a large number of questions in a timely manner to keep up with the flow of incoming posts on Stack Overflow. The tagging process should be efficient and not introduce significant delays in question visibility.

NFR-5	**Maintainability**	The system should be designed and implemented in a way that allows for easy maintenance and updates. The codebase should be modular and well documented, making it easier for developers to introduce enhancements or address issues
NFR-6	**Scalability**	The system should be able to handle an increasing volume of questions as Stack Overflow continues to grow. It should be designed to scale horizontally or vertically to accommodate the increasing workload without sacrificing performance

4. PROJECT DESIGN

1. Data Flow Diagrams

A Data Flow Diagram (DFD) for the autonomous tagging of Stack Overflow questions is a graphical representation that illustrates the flow of data and processes involved in the system responsible for automatically assigning tags to Stack Overflow questions. It provides a visual representation of how data is transformed and exchanged within the system.



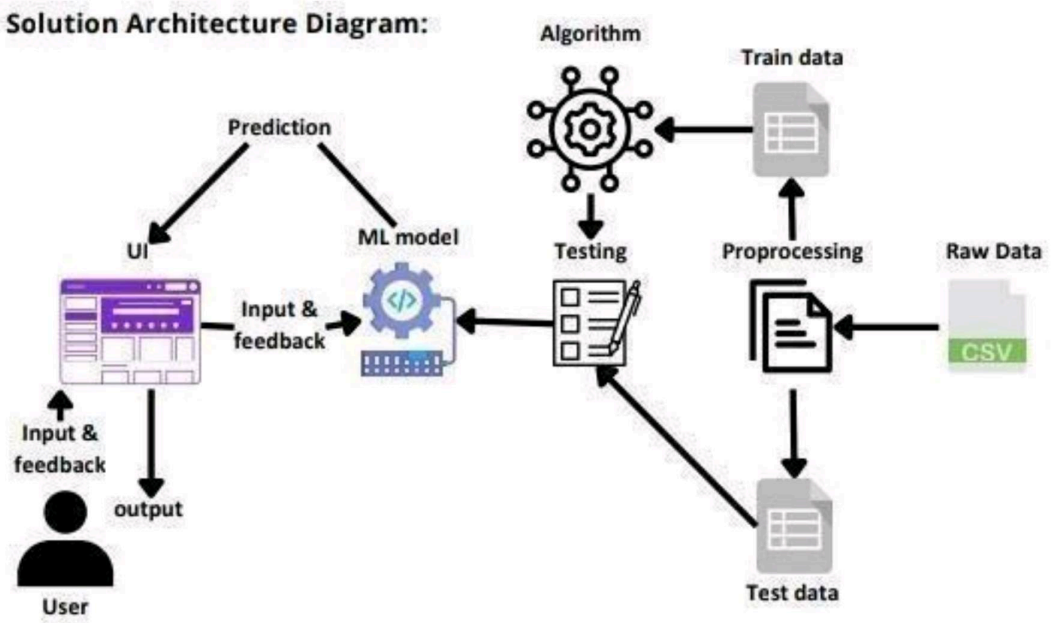
4.2 Solution Architecture :

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe The Structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.

- Provide specifications according to which the solution is defined,

managed, and delivered.



Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table 1 & table 2

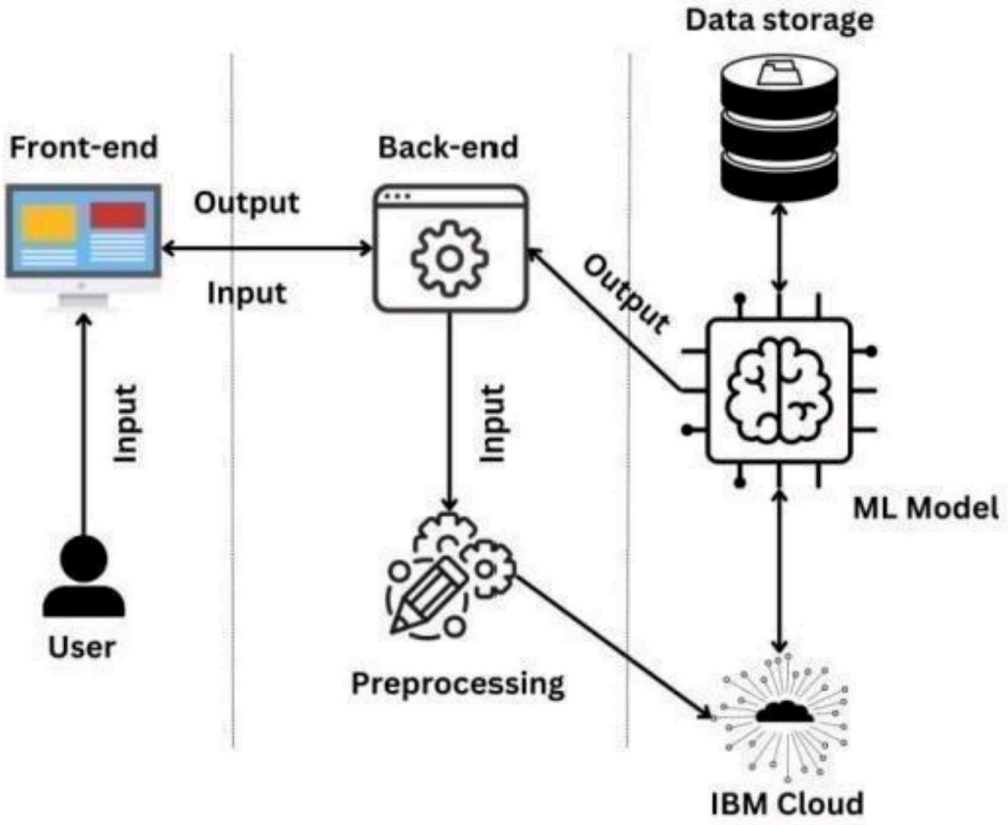


Table-1 : Components & Technologies:

S no.	Component	Description	Technology
1.	User Interface	How user interacts with application	HTML, CSS, JavaScript.
2.	Application Logic-1	Logic for Tag prediction and Classification	Python, Flask, scikit-learn.

3.	MachineLearning Model	Machine Learning Model for Tag prediction	Python, scikit-learn.
4.	Database	Data storage and retrieval	IBM DB2.
5.	External API-1	Purpose of External API used in the application	RESTful API.
6.	Infrastructure	Application Deployment and Hosting	IBM Cloud.

Table-2: Application Characteristics:

S no.	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask, scikit-learn.

2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Encryption.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	IBM Cloud Foundry.
4.	Availability	Justify the availability of (e.g. applications load use of servers balancers, distributed etc.)	IBM Cloud.

4.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement t (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Stack overflow user	User interface	USN-1	As a user, I want to ask a question.	Users can enter a question in the web interface.	High	Kishore

	Tag prediction	USN-2	As a user, I want an accurate Tag prediction.	Tags are predicted accurately based on the question.	High	Sathyavel
	Data processing and Post processing	USN-3	As a user, I want relevant and clean tags.	User input is pre-processed for cleaning and filtering.	Low	Vinoth Kumar
				Post-processing used to refine the predicted tags.	Low	Vinoth KUmar
	.Scalability and Performance	USN-4	As a user, I expect the platform to handle traffic.	Scalable architecture helps to handle the traffic.	Medium	Sivasanjay
	.Security and Access control	USN-5	As a user, I want my data to be secure.	Role based access control manages user permissions.	Low	Vinoth Kumar

5 CODING & SOLUTIONING

1. Packages

Packages can have a hierarchical structure, with subpackages and submodules. This allows for logical organization of code. For example, you can have a package called my_package, which can contain subpackages like my_package.subpackage1 and my_package.subpackage2.

+ Code + Text Connect

Importing required packages.

```
import pandas as pd
import numpy as np


import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import seaborn as sns

import warnings

import pickle
import time

import re
from bs4 import BeautifulSoup
import nltk
from nltk.tokenize import ToktokTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
from string import punctuation
```

Read The Datasets

Loading the dataset into the project.

```
df_tags = pd.read_csv("/content/drive/MyDrive/dataset_ibm/archive (2).zip (Unzipped Files)/Tags.csv", encoding="ISO-8859-1", dtype={'Tag': str})
df_question = pd.read_csv('/content/drive/MyDrive/dataset_ibm/archive (2).zip (Unzipped Files)/Questions.csv',encoding='ISO-8859-1')
```

```
print("question.csv shape: ",df_question.shape)
print("tags.csv shape: ",df_tags.shape)
```

```
question.csv shape:  (1264216, 7)
tags.csv shape:  (3750994, 2)
```

```
df_tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3750994 entries, 0 to 3750993
Data columns (total 2 columns):
 #   Column  Dtype
---  ------  -----
 0   Id      int64
 1   Tag     object
dtypes: int64(1), object(1)
memory usage: 57.2+ MB
```

5.2 Merging the dataset into new dataset

Merging the two dataset into one new dataset.

Consider column "id" as a common to merge the datasets.

```
df_tags['Tag'] = df_tags['Tag'].astype(str)
```

```
#grouping tags based on same ID"
grouped_tags = df_tags.groupby("Id")['Tag'].apply(lambda df_tags: ' '.join(df_tags))
```

```
grouped_tags.head()
```

```
Id
80                          flex actionscript-3 air
90       svn tortoisesvn branch branching-and-merging
120                             sql asp.net sitemap
180    algorithm language-agnostic colors color-space
260           c# .net scripting compiler-construction
Name: Tag, dtype: object
```

```
grouped_tags.reset_index()
```

	Id	Tag
0	80	flex actionscript-3 air
1	90	svn tortoisesvn branch branching-and-merging

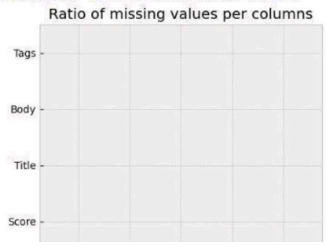
5.3 Data Cleaning

Data cleaning refers to the steps and techniques applied to raw data before it is fed into a machine learning model. It involves transforming and cleaning the data to make it suitable for the model to learn from and improve its performance. Data cleaning is a crucial step in the machine learning pipeline and can significantly impact the accuracy and effectiveness of the model.

Data cleaning.

```
plt.figure(figsize=(5, 5))
df_question_tags.isnull().mean(axis=0).plot.barh()
plt.title("Ratio of missing values per columns")
```

```
Text(0.5, 1.0, 'Ratio of missing values per columns')
```



5.4 Model Building

Model building refers to the process of creating a machine learning model that can make predictions or perform a specific task based on the provided data.

LDA

```
no_topics = 20
text = df_question_tags['Body']
```

```
vectorizer_train = TfidfVectorizer(analyzer = 'word',
                                   min_df=0.0,
                                   max_df = 1.0,
                                   strip_accents = None,
                                   encoding = 'utf-8',
                                   preprocessor=None,
                                   token_pattern=r"(?u)\S\S+", # Need to repeat token pattern
                                   max_features=1000)
```

```
TF_IDF_matrix = vectorizer_train.fit_transform(text)
```

```
lda = LatentDirichletAllocation(n_components=no_topics, max_iter=5, learning_method='online', learning_offset=50, random_state=11).fit(TF_IDF_matrix)
```

```
def display_topics(model, feature_names, no_top_words):
```

5.5 Predicted Output

```
C:\Users\Lenovo\Desktop\PROJECT-STACKOVERFLOW\flask>python model.py
Title    2
Body     1
Tags     0
dtype: int64
Title    0
Body     0
Tags     0
dtype: int64
Clf:  SGDClassifier
Jacard score: 47.518536109132164
Hamming loss: 0.9576505976410987
---
Clf:  LogisticRegression
Jacard score: 48.21895036808359
Hamming loss: 0.9754610939602628
---
Clf:  LinearSVC
Jacard score: 52.111005567429224
Hamming loss: 0.9660413203514604
---
```

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

```
mae = mean_absolute_error(y_test, pred)
mae
```

```
8.203867369235024
```

```
mse= mean_squared_error(y_test,pred)
mse
```

```
128.39806863056307
```

```
rmse = np.sqrt(mse)
rmse
```

```
11.331287156830996
```

```
r2_score(y_test,pred)
```

```
0.721307989574244
```

5.6 Website Creation

HTML

flask > templates > index.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>Stack Overflow Question Tagging</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Stack Overflow Question Tagging</h1>
        <form action="{{ url_for('predict') }}" method="POST">
            <label for="question">Enter your question:</label>
            <textarea id="question" name="question" rows="4" cols="50"></textarea>
            <input type="submit" value="Predict">
        </form>
        <div id="result">
            {% if result %}
            <p>Result: {{ result }}</p>
            {% endif %}
        </div>
    </div>
</body>
</html>

```

flask

flask > app.py

```python
#IMPORTING THE PACKAGES REQUIRED
from flask import Flask, request, jsonify, render_template
from tagpredictor import get_Tags

# Create the Flask application
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

# Define a route for model inference
@app.route('/predict', methods=['POST','GET'])
def predict():
    question=request.form['question']
    result = get_Tags(question)[0]
    return render_template('index.html', result=result)

if __name__ == '__main__':
    app.run(debug=True)

```

6. RESULT

1. Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Confusion Matrix (Based on tag ".net")-[[12033 54] [447 100]] , Accuray Score-81.27% & Classification Report(weighted avg) -[0.73,0.37,0.48,19887}	.net [[12033 54] [447 100]] ajax [[12526 15] [55 38]] algorithm [[12445 21] [87 81]] android [[11528 40] [194 872]] angularjs [[12443 7] [49 135]] Accuracy Score: 81.27% micro avg 0.78 0.37 0.51 19887 macro avg 0.61 0.25 0.33 19887 weighted avg 0.73 0.37 0.48 19887 samples avg 0.49 0.41 0.43 19887
2.	Tune the Model	Hyperparameter Tuning - GridSearchCV Validation Method - Cross Validation	GridSearchCV estimator: OneVsRestClassifier estimator: LinearSVC LinearSVC

7 ADVANTAGES & DISADVANTAGES

1. Advantages:

Efficiency: Autonomous tagging can significantly improve the efficiency of the question tagging process. Instead of relying solely on human moderators to manually assign tags, the system can automatically suggest relevant tags, saving time and effort.

Consistency: Autonomous tagging can help ensure consistency in the tagging process. Human moderators may have different interpretations or subjective biases when assigning tags, leading to inconsistencies. Automated tagging can provide a standardized approach, resulting in more consistent tags across questions.

Scalability: With the increasing volume of questions being posted on Stack Overflow, autonomous tagging can handle the scalability challenge. It can process a large number of questions in a relatively short period, allowing for timely and accurate tag assignments.

Improved user experience: By automatically suggesting relevant tags, autonomous tagging can enhance the user experience for both question askers and answers. It can help improve question visibility, attract relevant experts, and facilitate more accurate and targeted answers.

7.2 Disadvantages:

Limited accuracy: Autonomous tagging systems may not always accurately assign tags. They rely on machine learning algorithms trained on existing data, which may not capture the full complexity and nuances of new questions. This can result in incorrect or insufficient tags being assigned.

Lack of context: Autonomous tagging algorithms may struggle to grasp the full context of a question. They might miss subtle details or nuances that a human moderator could identify. Consequently, the assigned tags may not fully capture the essence of the question, making it harder for users to find the relevant content.

Evolving technology: Autonomous tagging systems require continuous updates and improvements to keep up with the ever-changing nature of programming languages, frameworks, and technologies. As new technologies emerge, the system may struggle to accurately tag questions related to these emerging domains.

Over-reliance on algorithms: Autonomous tagging should be used as a complementary tool rather than a replacement for human moderation. Over

Reliance on algorithms can result in a loss of human judgment and subjective interpretation, potentially leading to less nuanced or diverse tagging

8. CONCLUSION

In conclusion, autonomous tagging of Stack Overflow questions has both advantages and disadvantages. It offers efficiency, scalability, consistency, and improved user experience by automatically suggesting relevant tags for questions. It can save time and effort, ensure consistency in tag assignments, handle large volumes of questions, and attract relevant experts. However, it also has limitations such as limited accuracy, lack of context, dependency on evolving technology, and potential over-reliance on algorithms. Therefore, a balanced approach that combines automated tagging with human moderation is recommended to address these limitations and ensure the overall quality of tagged questions on Stack Overflow. By leveraging the strengths of both automation and human judgment, the tagging process can be optimized to enhance the user experience and provide more accurate and relevant content to the Stack Overflow community.

9. FUTURE SCOPE

The future scope of autonomous tagging of Stack Overflow questions is vast, with opportunities for advancements in accuracy, context awareness, adaptability, community involvement, multi-modal information processing, customization, and real-time capabilities. By continually pushing the boundaries of technology and incorporating user feedback, autonomous tagging can play a crucial role in improving the effectiveness and efficiency of the Stack Overflow platform.

10. APPENDIX

1. Source code

Model Building

```
#IMPORTING LIBRARIES
import pandas as pd
import numpy as np

#READ THE PROCESSED DATASET
df =
pd.read_csv('C:\\Users\\Lenovo\\Desktop\\PROJECT-STACKOVERFLOW\\dataset\\final_question_tags.csv')
print(df.isnull().sum())
df = df.dropna(subset=['Title','Body'])
print(df.isnull().sum())

#CONVERTING TAGS COLUMN INTO LIST
import ast
df['Tags'] = df['Tags'].apply(lambda x: ast.literal_eval(x))

#IMPORT ALL THE NECESSARY MODEL
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier
from scipy.sparse import hstack
from sklearn.metrics import hamming_loss
from sklearn.metrics import
confusion_matrix,accuracy_score,multilabel_confusion_matrix,classification_report


#ASSIGNING Y AS COLUMN TAGS,X1 AS BODY COLUMN,X2 AS TITLE COLUMN
X = df['Body']
y = df['Tags']

#CONVERT Y COLUMN TO CLASSES
multilabel = MultiLabelBinarizer()
y = multilabel.fit_transform(y)
multilabel.classes_
pd.DataFrame(y ,columns = multilabel.classes_)

#TF-IDF VECTORIZER
#tfidf = TfidfVectorizer(analyzer = 'word' , max_features = 10000,  ngram_range = (1,3), stop_words = 'english')
vectorizer_X1 = TfidfVectorizer(analyzer = 'word',
                                min_df=0.0,
                                max_df = 1.0,
                                strip_accents = None,
                                encoding = 'utf-8',
                                preprocessor=None,
```

```
                    token_pattern=r"(?u)\S\S+",
                    max_features=1000)



X_tfidf = vectorizer_X1.fit_transform(X)



#SPLITTING THE DATASET
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size = 0.2, random_state = 0)

#INITIALING THE MODELS
sgd = SGDClassifier()
lr = LogisticRegression()
svc = LinearSVC()



#JACCAD SCORE USED TO ANALYSIS THE ACCURACY OF THE MULTILABEL CLASSIFICATION  MODEL
def avg_jacard(y_true,y_pred):
    jacard = np.minimum(y_true,y_pred).sum(axis=1) / np.maximum(y_true,y_pred).sum(axis=1)
    return jacard.mean()*100

def print_score(y_pred, clf):
    print("Clf: ", clf._class.name_)
    print("Jaccard score: {}".format(avg_jacard(y_test, y_pred)))
    print("Hamming loss: {}".format(hamming_loss(y_pred, y_test)*100))
    print("---")
```

```
#PREPARING THE MODEL
for classifier in [sgd,lr,svc]:
    clf = OneVsRestClassifier(classifier)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print_score(y_pred, classifier)
    #y_pred = multilabel.inverse_transform(y_pred)

    # Calculate the accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy Score: {:.2f}%".format(accuracy * 100))

    # Generate the classification report
    report = classification_report(y_test, y_pred)
    print("Classification Report:")
    print(report)

#EXPORTING THE BEST MODEL
import joblib
joblib_file = "tagPredictor.pkl"
joblib.dump(clf, joblib_file)
```

Tag Prediction Function

```
#IMPORTING PACKAGES REQUIRED
import  joblib
import  pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.preprocessing import MultiLabelBinarizer

from scipy.sparse import hstack


#LOADING THE DATASET

df =
pd.read_csv('C:\\Users\\Lenovo\\Desktop\\PROJECT-STACKOVERFLOW\\dataset\\final_question_tags.csv')

df = df.dropna()


#CONVERTING TAGS COLUMN INTO LIST

import ast

df['Tags'] = df['Tags'].apply(lambda x: ast.literal_eval(x))


#LOADING THE PREDICTION FILE

tag_pred = joblib.load('tagPredictor.pkl')


#iNTIALISATION OF THE REQUIRED METHODS

vectorizer = TfidfVectorizer(analyzer = 'word',
                             min_df=0.0,
                             max_df = 1.0,
                             strip_accents = None,
                             encoding = 'utf-8',
                             preprocessor=None,
                             token_pattern=r"(?u)\S\S+",
                             max_features=1000)
```

```
x_tfidf = vectorizer.fit_transform(df['Body'])



multilabel = MultiLabelBinarizer()
y = multilabel.fit_transform(df['Tags'])

#DEFINING THE FUNCTION THE TAG PREDICTION
def get_Tags(question):
    question = [question]
    question = vectorizer.transform(question)
    tags = multilabel.inverse_transform(tag_pred.predict(question))
    return tags

out = get_Tags("what is python and how it works")
print(out)
```

Flask

```
#IMPORTING THE PACKAGES REQUIRED
from flask import Flask, request, jsonify, render_template
from tagpredictor import get_Tags

# Create the Flask application
app = Flask(_name_)
```

```
@app.route('/')
def index():
    return render_template('index.html')


# Define a route for model inference
@app.route('/predict', methods=['POST','GET'])
def predict():
    question=request.form['question']
    result = get_Tags(question)[0]
    return render_template('index.html', result=result)


if _name_ == '_main_':
    app.run(debug=True)
```

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Stack Overflow Question Tagging</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Stack Overflow Question Tagging</h1>
        <form action="{{ url_for('predict') }}" method="POST">
            <label for="question">Enter your question:</label>
            <textarea id="question" name="question" rows="4" cols="50"></textarea>
```

```
        <input type="submit" value="Predict">
    </form>
    <div id="result">
        {% if result %}
        <p>Result: {{ result }}</p>
        {% endif %}
    </div>
  </div>
</body>
</html>
```

10.2 GITHUB & DEMO LINK

GITHUB: https://github.com/GokulRR001/microsoft-intern

DEMO LINK:
https://drive.google.com/file/d/1EqjS1shqRTQ2Ia4BFL9BDqeM6TqB_JJi/view?usp=drivesdk