Auto-Tagging Stack Overflow Questions

Pratheek Bagepalli Ravi Prakash
UIN: 334003615

Advisor
Duncan M. Walker

Texas A&M University

1. Abstract

Stack Overflow is one of the most important websites where programmers share knowledge, get coding problems solved, and learn from one another's experience. Yet, the manual question tagging that is most important for discoverability and organization is subject to variation. Mislabeling or under-labeling of questions by users discourages both content retrieval and community interaction. To solve this problem, we present a machine learning-based multi-label classification system capable of auto-tagging new Stack Overflow questions.

The proposed solution employs TF-IDF feature extraction and Support Vector Machines (SVM) with One-vs-Rest classification. Accuracies of various classifiers such as SVM, Naive Bayes, Logistic Regression, and Random Forest were tried on various sets of features (title, body, and both). The best model with both title and body as input and SVM achieved an F1-score of 0.7959 and accuracy of 0.6801. This model was integrated into a lean Flask web application with real-time tag prediction for new questions. The practical implications of this system are substantial: it greatly enhances question findability and user experience by lowering the cognitive burden on the user and ensuring tag consistency throughout the site

2. Objectives

The primary goal of this study is to automate the prediction of tags for Stack Overflow questions using machine learning. This involves creating an effective multi-label classification model, contrasting different algorithmic approaches, and deploying the best-performing model in an end-user application.

- **Develop a multi-label classifier for Stack Overflow tags**: Because Stack Overflow questions tend to inquire about several categories, the objective is to create a model that predicts multiple appropriate tags for each question, given its title and body.
- **Compare SVM, Naive Bayes, Logistic Regression, Random Forest performance**

- **Integrate the best model into a Flask web app for real-time predictions:** In addition to conducting experiments, the aim encompasses the implementation of the model within a user-friendly web interface that permits individuals to enter a query and promptly obtain recommended tags.

3. Proposed Methodology

3.1 Data Acquisition

A Stack Overflow data dump was obtained using Google BigQuery. This contained questions, answers and corresponding tags. The process began by restricting the tag range to the top 50 most frequent tags, but encountered a highly skewed distribution: the most frequent tag had over 45,000 questions, and the least frequent had just 171. To even out this imbalance, the range was restricted to the top 20 tags. This gave a more balanced distribution and a final dataset of 760 examples ready to train and test.

3.2 Text Preprocessing

Each question has a title and a body section (in HTML). The HTML tags, web URLs, and punctuation were stripped. Stopwords were also removed, and the Snowball Stemmer was applied to stem words to their base form. This step provides semantic consistency and dimensionality reduction.

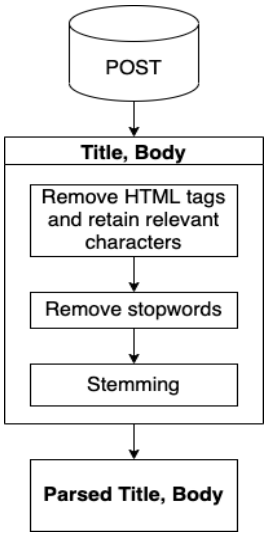

Figure 1: Text preprocessing pipeline

3.3 Feature Extraction

The text data was transformed into numerical representations via TF-IDF with unigrams and bigrams:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t)$$

Where,

TF(t, d): Frequency of term t in document d

IDF(t): Inverse document frequency of term t

This approach emphasizes rare but important terms and diminishes the impact of common, less discriminative phrases.

3.4 Model Architecture

A One-vs-Rest (OvR) strategy to handle the multi-label nature of the problem, where each question can belong to multiple tags. OvR is well-suited for this scenario as it allows training a separate binary classifier for each tag, enabling the model to learn independent decision boundaries for each label. Below is a performance summary:

Classifier	**Feature Set**	**F1-Score**	**Accuracy**
Linear SVM	Title + Body	0.7959	0.6801
Naive Bayes	Title + Body	0.5415	0.4851
Logistic Regression	Title + Body	0.6618	0.5736
Random Forest	Title + Body	0.5980	0.4880

Table 1: Performance Evaluation of different models

SVM outperformed all other models, especially on the combined title + body input. This is probably because they excel in handling high-dimensional, sparse data, a common property of text classification. SVMs are also better at finding optimal hyperplanes for separation, which helps in discriminating between subtle contextual differences between tags, especially when combined with expressive TF-IDF features.

3.4 System Design

The architecture of the automated tag prediction system is designed to be modular, efficient, and suitable for real-time interaction. It consists of five stages that process the input question and output predicted tags. The system is deployed as a Flask web application, making it accessible through a simple user interface.

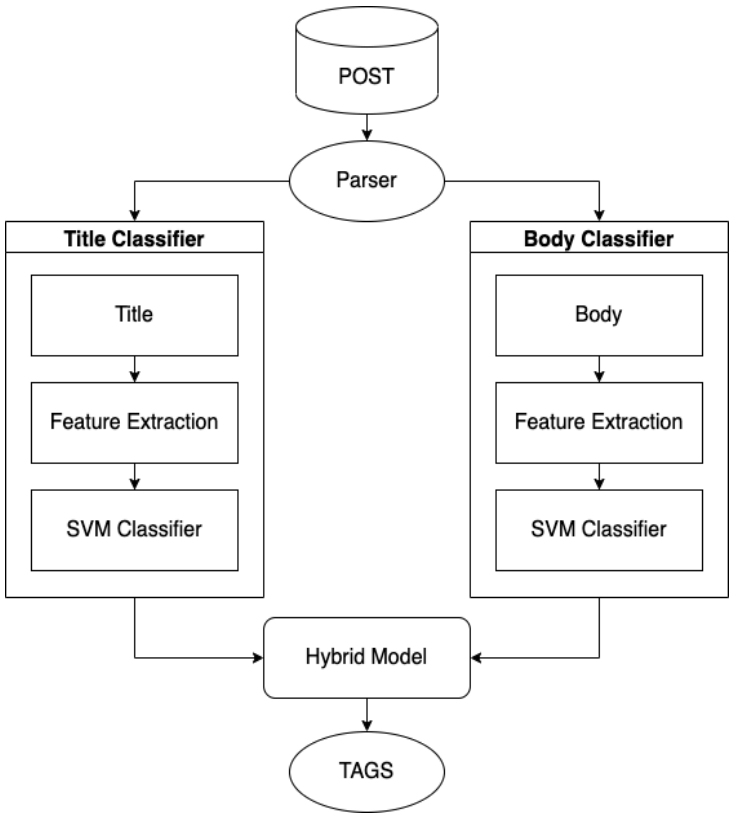

Figure 2: System design of Flask Application

1. User Input : Users provide a question consisting of a title and a body, similar to how questions are posted on Stack Overflow.

2. Text Cleaning: The input text undergoes preprocessing, where HTML tags, special characters, and URLs are removed. This is followed by stopword removal and stemming using the Snowball Stemmer, ensuring consistency and reducing dimensionality in the text data.

3. TF-IDF Vectorization: The cleaned text (title and body) is converted into numerical feature vectors using TF-IDF (Term Frequency-Inverse Document Frequency) with n-grams. This step captures both word importance and local context by accounting for word combinations.

4. SVM Prediction: The TF-IDF vectors are fed into a pre-trained One-vs-Rest Linear SVM classifier, which outputs a set of tags relevant to the content. Each binary classifier in the OvR scheme independently decides whether a specific tag should be assigned.

5. Tag Output: The predicted tags are returned to the user through the web interface, providing immediate, relevant suggestions to help categorize the question accurately.

4. Results

A 10-fold cross-validation was conducted to assess model consistency. The best model, SVM with title + body achieved: **F1-Score: 0.7959 & Accuracy: 0.6801**

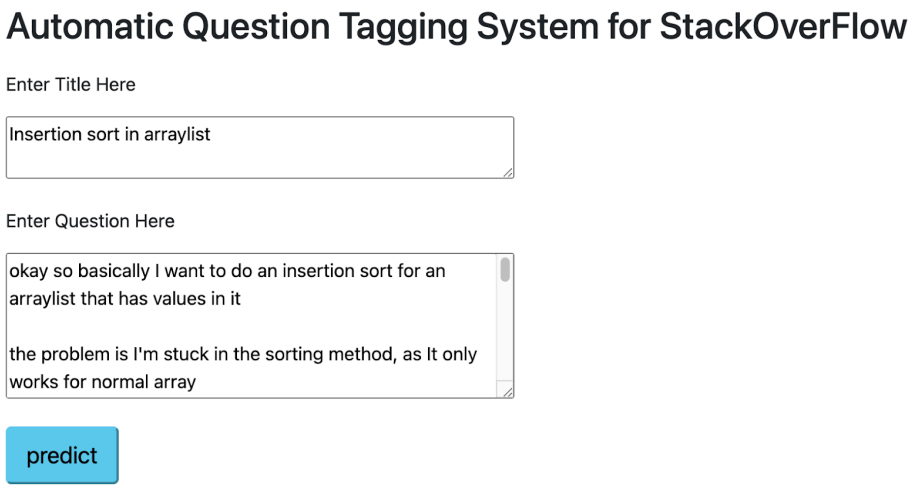

Figure 3: Page for user to enter their post

Figure 4: Output of the model

5. Future Work

As part of future work we plan on adding a programming language classifier to the existing system in order to make use of code snippets present in the certain question as it might turn out to be valuable data which could be used to make classification. We also plan to increase the support to the top 75 tags but this would need a larger dataset as not many instances of less popular tags are present.

References:

[1] Advanced Automated Tagging for Stack Overflow: A Multi-Stage Approach Using Deep Learning and NLP Techniques | IEEE Conference Publication | IEEE Xplore

[2] Post2Vec: Learning Distributed Representations of Stack Overflow Posts | IEEE Journals & Magazine | IEEE Xplore

[3] Automatic Question Tagging with Deep Neural Networks | IEEE Journals & Magazine | IEEE Xplore

[4] Unsupervised extreme multi label classification of stack overflow posts | Proceedings of the 1st International Workshop on Natural Language-based Software Engineering

[5] SOTagger - Towards Classifying Stack Overflow Posts through Contextual Tagging (S)

[6] Predicting tags for stack overflow questions using different classifiers | IEEE Conference Publication | IEEE Xplore

[7] Stack Exchange Data Explorer