

Automated Text Classification

Ziqi Zhai

**The University of Adelaide**

4533\_COMP\_SCI\_7417\_7717 Applied Natural Language Processing

Lecturer: Dr. Orvila Sarker

**Table of Contents**

[1. Abstract 2](#_Toc196193965)

[2. Introduction 2](#_Toc196193966)

[2.1. Data collection 2](#_Toc196193967)

[3. Preprocessing 4](#_Toc196193968)

[3.1. Remove HTML and image 4](#_Toc196193969)

[3.2. Punctuation Removal 4](#_Toc196193970)

[3.3. Lowercasing 4](#_Toc196193971)

[3.4. Whitespace Normalization 5](#_Toc196193972)

[3.5. Stopword Removal and Lemmatization 5](#_Toc196193973)

[3.6. Tracking and saving 5](#_Toc196193974)

[4. Data Visualisation 5](#_Toc196193975)

[4.1. Word cloud and frequent 5](#_Toc196193976)

[4.2. Bi-grams and Tri-grams 6](#_Toc196193977)

[5. Implementation Details 8](#_Toc196193978)

[5.1. TF-IDF 8](#_Toc196193979)

[5.2. MultiLabelBinarizer 8](#_Toc196193980)

[5.3. One-VS-Rest 8](#_Toc196193981)

[5.4. Training and evaluation 8](#_Toc196193982)

[5.5. Result 9](#_Toc196193983)

[5.6. Discussion 10](#_Toc196193984)

[6. Conclusion 10](#_Toc196193985)

[7. References 11](#_Toc196193986)

# Abstract

This project aims to build a multi-label classification model to categorize Natural Language Processing (NLP)-related questions on Stack Overflow based on their tags. Using real-world question-and-answer data scraped from Stack Overflow, the input features are derived from the text information in the title, body, and accepted answer after text preprocessing. The TF-IDF vectorization method is employed to convert the text into numerical features, and multi-label binarization is applied to process the tags. A One-vs-Rest logistic regression model is used for training.

The model's effectiveness in real-world scenarios is validated by evaluating metrics such as precision, recall, F1-score, and support (Singh, 2024) for the multi-label task. The experimental results demonstrate that this method achieves reasonable accuracy in identifying mainstream NLP tags, providing foundational technical support for building a knowledge classification system for Natural Language Processing.

# Introduction

Programming forums have become vital platforms for sharing knowledge, seeking solutions, and collaboratively addressing coding challenges. These forums host millions of questions and answers contributed by developers and learners worldwide, serving as a rich source of technical data. Understanding these platforms' operational dynamics and user engagement patterns, such as which types of posts receive responses, question view frequencies, and correlations between specific tags and activity, which can provide valuable insights for enhancing the user experience.

In this report, I analyse a dataset of Natural Language Processing (NLP)-related posts from Stack Overflow. The dataset includes post titles, content, view counts, accepted answers, and user-generated tags, among other metadata.

The objectives in this project are:

* Visualizing post activity to explore community engagement trends.
* Building a multi-label classification model capable of predicting relevant tags for a given question based on its title, content, and answer text.
* Evaluating model performance using standard metrics.

In this project, developing a text-based multi-label classification model, by using NLP-related questions on Stack Overflow as training data, so that the model can predict the appropriate labels for posts. The results can inform future automated question labelling and precise positioning of questions based on keywords.

## Data collection

The data will be collected from StackExchange, comprising a dataset of over 20,000 entries. These entries consist of NLP-related posts from Stack Overflow, including key components such as: “Title”, “Body”, “Tags” and “AcceptedAnswerBody”.

This dataset will serve as the foundation for training and evaluating the multi-label classification model.

Data download: https://data.stackexchange.com/stackoverflow/queries?q=nlp

|  |
| --- |
|  |
| Figure 1: Post count |

|  |
| --- |
|  |
| Figure 2: Answer view count |

Figure 1 shows the response status of NLP-related posts on Stack Overflow. Out of a total of 20,430 posts, only 8,509 were answered, accounting for 41.65% of the total, while 11,921 posts remained unanswered, making up 58.35% of the total. This indicates that in this dataset, the majority of posts received no answers.

Figure 2 illustrates the relationship between the number of answers and the number of views. The red trendline suggests a positive correlation between the number of answers and the number of views. The vast majority of posts have fewer than 10 answers and fewer than 100,000 views. The upper-right area shows a small number of posts that received an exceptionally high number of answers and views.

Although the regression line shows an upward trend, the distribution of the points—particularly some posts with over 100,000 views but only one or two answers, as well as some with more than 10 answers (even exceeding 50) but relatively low view counts—suggests that the number of answers and the number of views are not strongly linked, other factors such as popular tags, title content, and posting time, may also influence the view count.

# Preprocessing

Collected text data often contains a lot of noise, including HTML tags, embedded images, punctuation, inconsistent case, irregular whitespace, and stop words. Before feeding this data into a machine learning model for training, it needs to be preprocessed, which improves the quality of the input data when the model is trained.

Preprocessing makes the format of the cluttered raw text data uniform, which helps in better feature extraction. In this project, the title, body and accepted answers of each post were preprocessed with HTML and image removal, special character handling, lowercase, whitespace normalisation, stop word removal and lemmatisation. These steps can be of great help for subsequent multi-label classification tasks.

## Remove HTML and image

Many text data sources, such as online forums or web-scraped content, include HTML tags and embedded images. These elements do not contribute to the semantic understanding of the text and may introduce noise into the model. Therefore, we parse the raw HTML content using a library like BeautifulSoup to extract only the textual information while removing all HTML tags. Additionally, any <img> tags or embedded images are explicitly removed to prevent non-textual content from affecting downstream natural language processing tasks. This step ensures a cleaner, more consistent text input for model training.

## Punctuation Removal

These characters—such as commas, periods, question marks, and symbols often do not carry meaningful semantic information in the text classification tasks. Using space to replace all punctuation characters can help to reduce noise in the dataset and ensures that tokens are clean and consistent, which focus the learning process on meaningful words rather than irrelevant symbols.

## Lowercasing

In NLP, converting all letters to lowercase such us transforming "Letter", "letter", and "LETTER" into the same word "letter", it helps reduce vocabulary size and improves the performance and efficiency of machine learning models. This approach is particularly effective when letter case is not semantically important for the task.

## Whitespace Normalization

Text data often contains various whitespace characters, which typically originate from user input spacing or as a result of replacing special symbols—especially after removing HTML tags and special characters. Handling whitespace ensures that the input text is clean and consistent, which is crucial for effective text analysis and modeling. This processing significantly reduces the potential negative impact on model performance.

## Stopword Removal and Lemmatization

To enhance the quality of the textual features, the preprocessing pipeline includes both stopword removal and lemmatization. Stopwords are commonly used words such as “the,” “is,” or “and” that generally do not contribute significant meaning in natural language processing tasks. Removing these words reduces noise and improves the efficiency of the model.

Lemmatization is applied to reduce each word to its base form, it ensures that resulting tokens are valid words. This process helps normalize variations of words such as “running,” “ran,” and “runs” all become “run”, which allow the model to treat them as a single feature and thereby improving generalization of the model.

## Tracking and saving

During data preprocessing, the tqdm library can be used to monitor progress. It shows a progress bar that helps developers track the data processing workflow, which is especially useful when working with large datasets.

After completing the preprocessing step, the cleaned dataset needs to be saved, which avoids reprocessing the original data each time. In this project, the cleaned data is saved as a CSV file named ‘cleaned\_data.csv’, which includes the processed headers, body, and accepted answer content.

# Data Visualisation

## Word cloud and frequent

Figure 3 and 4 show the word cloud figure and word frequency figure, in text data analysis, word clouds and words frequency figure are two common visualization methods. They have the same goal to help identify the most frequent keywords in the dataset. In the word cloud figure, we can intuitively observe that terms such as "example", "result", "S", "py" and "use" appear in larger fonts, indicating their high frequency and giving an overall impression that the discussions are centered around examples, results, and usage. In the word frequency figure offers a more precise view with specific numbers — words like "word", "text", "model", and "sentence" occur over 20,000 times, highlighting their prominence in the dataset.

Whilst word clouds can be visually appealing, word frequency figure can more directly show the difference in words frequency used. Using two figures together can often lead to a more intuitive understanding (Wang, 2022).

|  |
| --- |
|  |
| Figure 3: Word cloud |
|  |
| Figure 4: Top 20 most frequent words |

## Bi-grams and Tri-grams

Figure 5 and 6 illustrates the top 20 most frequent bi-grams and tri-grams, which found in the text data of questions and accepted answers. Common bi-grams such as "site package," "train model", and "batch size", tri-grams such as "natural language processing" and "en core web", show topics about NLP workflows and machine learning practices.

The presence of the phrases ‘stanford nlp’ and ‘edu stanford nlp’ indicate that academic resources about Stanford are cited by many people.

The recurrence of bi-grams such as "word word", "nan nan" and "nan nan nan" may point to either repeated placeholder terms or noise in the data.

Overall, this visualization reveals key phrase patterns and common technical topics discussed by users, especially around tools, models, and preprocessing techniques (Ashokcharan.com, 2023).

|  |
| --- |
|  |
| Figure 5: Top 20 Bi-grams |
|  |
| Figure 6: Top 20 Tri-grams |

# Implementation Details

## TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is used to extract textual features. It is calculated by multiplying Term Frequency (TF) and Inverse Document Frequency (IDF). TF refers to how frequently a word appears in a single document, while IDF measures how commonly that word appears across the entire corpus. The resulting TF-IDF value represents the importance of a term to a particular document within the corpus (Karabiber, 2024). If a term appears frequently in a specific document but rarely across the corpus, it indicates that the word is highly important for that document. TF-IDF transforms text into numerical vectors, making it easier for models to interpret and understand the data.

## MultiLabelBinarizer

MultiLabelBinarizer can convert multi-label data into one-hot encoded format, where columns represent the labels and rows represent the instances. Value 1 indicates the presence of a label and 0 indicates its absence (Admin, 2023). This provides a more suitable data format for models in multi-label classification tasks.

## One-VS-Rest

The One-vs-Rest is a classification strategy that transforms a multi-class or multi-label classification problem into multiple binary classification problems. For a dataset with N possible labels, One-vs-Rest trains N separate classifiers. Each classifier is responsible for distinguishing whether a sample belongs to a particular class or not. In multi-label classification(Glare, n.d.), this approach allows a single instance to be assigned multiple labels. Each classifier outputs a binary decision, and the final prediction is a combination of all positive outputs.

## Training and evaluation

To train and evaluate the classification model effectively, the dataset need to split into a training set and a test set using an 80/20 ratio. The training set was used to train the model, while the test set was reserved for final evaluation.

5-fold cross-validation strategy was applied on the training set to evaluate model performance during training. This splits the training data into five subsets, where the model is trained on four sets and validated on the remaining set in rotation(Ebner, 2023). The average F1-micro score across all folds was calculated to evaluate the model’s effectiveness in training.

The training model chose is a combination of TF-IDF, One-vs-Rest, and Logistic Regression. One-vs-Rest is particularly effective when combined with classifiers that are naturally binary, such as Logistic Regression(Brownlee, 2020), making it a popular choice in multi-label text classification tasks.

After cross-validation, the model was retrained on the full training set and used to predict labels on the test set. Performance metrics precision, recall, F1 score and support were used to evaluate the model's performance in the multi-label classification task.

## Result

Figure 7 shows the evaluation metrics demonstrate the model’s performance in different tags in the multi-label classification task. The model achieves high precision across nearly all tags, which means it rarely assigns incorrect labels. For recall, the model performs strongly on common tags like "nlp", "nltk", and "spacy", with recall scores above 0.65, suggesting reliable identification of those categories. In contrast, tags such as "lemmatization", "transformer", and "text-classification" have low recall from 0.14 to 0.28, which indicating the model often fails to detect these labels when they are actually present.

The samples-averaged scores are relatively low (F1 = 0.34), which is expected in multi-label tasks, as the strictness of this metric penalizes partial label matches for each sample.

Overall, the model demonstrates a high success rate in identifying labels, but tends to be conservative in assigning them, resulting in lower recall for certain categories. This trade-off suggests that there is room for improvement in recognizing less frequent or semantically similar tags.

|  |
| --- |
|  |
| Figure 7: Model evaluation |

## Discussion

To improve the model’s performance in the future, deep learning techniques can be considered, particularly by using pre-trained models like BERT. Unlike traditional TF-IDF features, BERT can capture rich contextual and semantic information from text, enabling the model to better distinguish between semantically similar tags and more effectively handle low-frequency labels (Devlin et al., 2018). This approach is expected to achieve higher recall and overall performance, especially in situations where traditional vectorization methods fall short. By leveraging BERT, the model's ability to understand language patterns will be enhanced, allowing it to make more informed predictions.

# Conclusion

This project focuses on multi-label classification of posts about NLP on Stack Overflow. After text preprocessing, I trained a Logistic Regression model under the One-vs-Rest strategy. The results show that the model performs well in terms of precision, but the recall is relatively moderate, especially when dealing with low-frequency or semantically similar tags. Although the model demonstrates strong performance as a baseline approach, future improvements could involve incorporating advanced techniques such as BERT-based contextual embeddings to better capture deep semantic relationships and enhance classification performance. This study not only confirms the effectiveness of traditional machine learning methods for this task but also highlights the potential of leveraging pre-trained language models to optimize results.

# References

1. Admin (2023). Scikit-Learn’s preprocessing.MultiLabelBinarizer in Python (with Examples) | PythonProg. [online] PythonProg. Available at: https://www.pythonprog.com/sklearn-preprocessing-multilabelbinarizer/ [Accessed 18 Apr. 2025].
2. Ashokcharan.com. (2023). MarketingMind. [online] Available at: https://www.ashokcharan.com/Marketing-Analytics/~sma-n-grams-bigrams-and-trigrams.php#gsc.tab=0 [Accessed 21 Apr. 2025].
3. Brownlee, J. (2020). How to Use One-vs-Rest and One-vs-One for Multi-Class Classification. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/ [Accessed 21 Apr. 2025].
4. Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.04805 [Accessed 21 Apr. 2025].
5. Ebner, J. (2023). Cross Validation, Explained - Sharp Sight. [online] www.sharpsightlabs.com. Available at: https://www.sharpsightlabs.com/blog/cross-validation-explained/ [Accessed 21 Apr. 2025].
6. Glare, T. (n.d.). Essential Data Science Tips: How to Use One-Vs-Rest and One-Vs-One for Multi-Class Classification. [online] KDnuggets. Available at: https://www.kdnuggets.com/2020/08/one-vs-rest-one-multi-class-classification.html [Accessed 21 Apr. 2025].
7. Karabiber, F. (2024). TF-IDF — Term Frequency-Inverse Document Frequency – LearnDataSci. [online] www.learndatasci.com. Available at: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ [Accessed 21 Apr. 2025].
8. Singh, D. (2024). Building a Multi-Label Multi-Class Text Classifier with BERT: A Step-by-Step Guide with Code. [online] Medium. Available at: https://medium.com/ai-enthusiast/building-a-multi-label-text-classifier-with-bert-a-step-by-step-guide-with-code-2bbeba8bd85a [Accessed 19 Apr. 2025].
9. Wang, M. (2022). *WordClouds with Python*. [online] Medium. Available at: https://medium.com/@m3redithw/wordclouds-with-python-c287887acc8b [Accessed 20 Apr. 2025].