

Automated Text Classification

Individual :

Name: Sadman Sharif

ID: A1944825

**The University of Adelaide**

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

Lecturer: Dr. Orvila Sarker

**Table of Contents**

[1. Abstract 2](#_Toc162337384)

[2. Introduction 2](#_Toc162337385)

3.Dataset Collection……………………………………………………………………………………………………………….2

[4. Dataset visualization 3](#_Toc162337392)

5.Dataset Pre-processing…………………………………………………………………………..5

6. Pre-processed Data Analysis……………………………………………………………………5

7. Rule-Based Categorization……………………………………………………..………………8

8. BERT-Based Classification…………………………………………….………………………9

9. Fine-tuning Bert Model………………………………………………………..………………9

10. Comparative Evaluation………………………………………………….…………………10

11.Conclusion …………………………………………………………………………………….12

# Abstract

This project addresses the challenge of automatically classifying developer queries on natural language processing (NLP) uploaded on Stack Overflow. Our main goal is to create a disciplined knowledge base covering typical implementation problems, conceptual ambiguities, and task-specific difficulties experienced by developers. Using the StackExchange API, we compiled a dataset of more than 28,000 posts and applied a dual approach: (i) a rule-based heuristic categorization system anchored in language signals, and (ii) a transformer-based classifier using BERT fine-tuning. The BERT paradigm guarantees scalability; the rule-based approach guarantees explainability. Evaluation was done against a 100 question manually annotated dataset. Despite data imbalance, our BERT model attained a validation accuracy of 68.24%; real-world accuracy on unseen labels was lower. This paper emphasizes the trade-offs and strengths of mixing interpretable and learning-based models for NLP developer support systems.

# Introduction

Natural language processing keeps growing in commercial use as well as academic study. The complexity of libraries, tools, and activities increases along with the difficulties experienced by practitioners. Although developers often consult Stack Overflow for direction, many entries remain unstructured and challenging to search methodically. Organizing such postings into relevant categories will help tool builders, teachers, and students all around.   
This project aims to classify Stack Overflow NLP-related developer questions into a set of logical, non-overlapping groups. We investigate two complimentary techniques: a machine learning approach using BERT for semantic comprehension and rule-based categorization using hand-crafted language patterns. Posts, tags, and approved responses abound in the Stack Overflow dataset gathered via StackExchange API. We evaluate each technique using both qualitative analysis and statistical evaluation.

**3. Dataset Collection**

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Total posts | 28,041 |
| Posts with accepted answers | 11,162 |
| Unanswered posts | ~6,100 |
| Median time to accepted answer | 0.85 days |
| Average view count per post | ~980 |
|  |  |

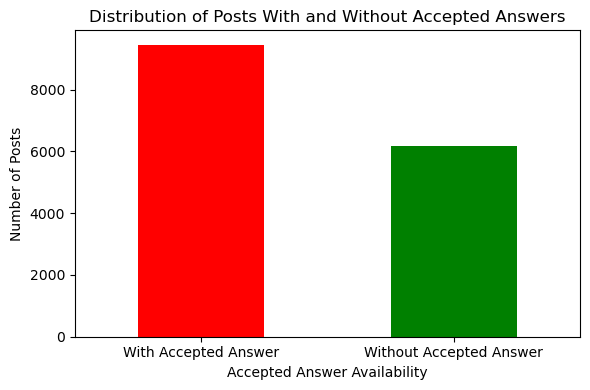
We gathered posts tagged with [nlp] along with associated tags including spacy, transformers, huggingface, and tokenizing using the StackExchange API to guarantee wide coverage and dataset diversity. 28,041 entries in all were gathered; each included the title, complete description, tags, view count, and, if possible, approved response.

**Dataset Sources and API Links**

To collect the dataset, we used the StackExchange API to gather NLP-related posts from Stack Overflow. Below are the key resources and API links referenced during data collection:

* [Stack Exchange API (official documentation)](https://api.stackexchange.com/docs)
* [Stack Overflow tag: NLP](https://stackoverflow.com/questions/tagged/nlp)
* [StackAPI Python wrapper (used for querying)](https://github.com/AWegnerGitHub/stackapi)

**4.Datasset visualization**

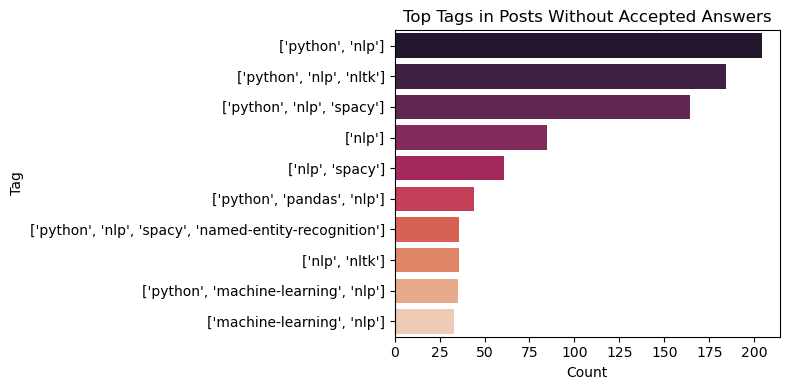


Of more than 15,000 posts examined, most (~9,400) had acceptable responses, suggesting great community involvement; about 6,100 remain unresolved.

A diagram of a distribution of a view

AI-generated content may be incorrect.

Most Stack Overflow entries in our sample lie between 100 and 1000 views; the red dashed line denotes the median at roughly 600 views, hence stressing modest community exposure.



From the bar plot, we can observe that combinations like ['python', 'nlp'], ['python, 'nlp', 'nltk'], and ['python, 'nlp','spacy'] predominate, implying that many unanswered queries center on basic toolchains and integration concerns in NLP pipelines.

A screenshot of a computer

AI-generated content may be incorrect.

Strong co-occurrence across fundamental NLP tags including nlp, Python, and nltk suggests typical usage patterns from our heatmap. This proves that most developers' questions center on basic tools and libraries working in concert.

**5.Dataset pre- preprocessing**

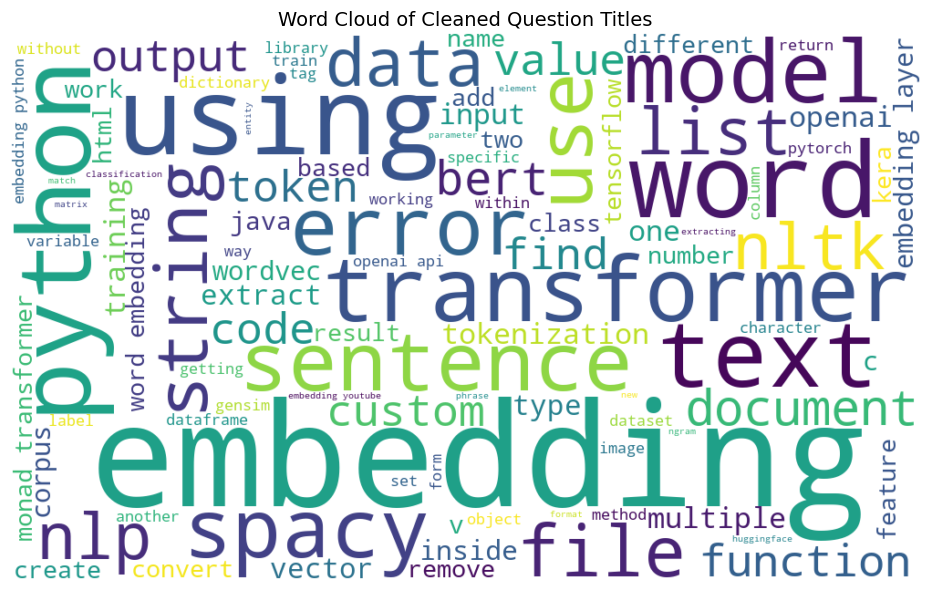
To prepare the text for rule-based filtering and transformer-based modeling, we applied several preprocessing steps:

|  |  |
| --- | --- |
| * **Preprocessing Step** | * **Method Used** |
| * HTML Tag Removal | * BeautifulSoup |
| * Lowercasing | * All text normalized to lowercase |
| * Special Character & Number Filtering | * Regex-based filtering |
| * Tokenization | * NLTK tokenizer |
| * Stopword Removal | * NLTK English stopword list |
| * Lemmatization | * WordNet Lemmatizer (from NLTK) |

Following preprocessing, we created two fresh fields—clean\_title and clean\_answer—per item that provided targets for additional classification. While lowering formatting and markup's noise, these cleaned fields maintain linguistic sense.

**6. Pre-processed Data Visualization**

We used visual analytics to better understand our dataset:

****

The most often occurring terms from cleaned question titles linked to NLP are shown in the word cloud, which highlights prevalent themes like "embedding,," "python, "transformer, "model,," and "text,," thereby implying a significant focus on implementation and text processing tasks.

A graph showing a number of different colored bars

AI-generated content may be incorrect.

top 20 most frequent tags in NLP-related Stack Overflow posts, with **"nlp"**, **"python"**, and **"nltk"** emerging as the most dominant, reflecting the central role of Python-based tools and libraries in natural language processing discussions.

**7. Rule-Based Categorization**

Nine categories were defined via sentence-level pattern matching across clean\_title + clean\_answer:

|  |  |
| --- | --- |
| Category Name | Pattern Examples |
| Implementation Issue | "how to", "how do I", "configure" |
| Task - Classification | "text classification", "binary classification" |
| Task - Tokenization | "tokenize", "word tokenizer" |
| Task - Similarity | "cosine similarity", "semantic match" |
| Task - Lemmatization | "stemming", "lemmatize" |
| Task - Language Identification | "detect language", "language of text" |
| Understanding Issue | "what is", "what does", "explain" |
| Library Specific | Mentions of spacy, nltk, transformers |
| Deployment/Serving Issue | "deploy", "serve model", "production" |

### ****Definition of the Categories****

#### **1. Issue with Implementation**

Questions in this area revolve on fixing programming mistakes, integration challenges, or unanticipated behavior in natural language processing systems. Usually beginning with "how to," "why is this failing," or "what is the correct way to..." these blogs center troubleshooting chores.  
**Example:** "How would one load a custom model in spaCy?"

#### **2. Library Specific**

Covers explicitly targeted NLP libraries and their use, including spaCy, NLTK, Hugging Face Transformers, or Gensim, explicitly. Posts in this category usually address unique to a given tool or framework defect, documentation confusion, or integration issues.

#### **3. Issue of Deployment and Serving**

This topic covers questions about model inference via APIs, containerization (e.g., using Docker), and performance issues for providing NLP models into production systems. These blogs are valuable since they frequently cross NLP and MLOps.

#### **4. Work: Tokenizing**

Contains entries on the division of raw text into tokens, words, or subword units. These might ask about tokenizer choice, customizing, or managing edge situations including punctuation or special characters.

#### **5. Work - Similarity**

Covers works with semantic or syntactic similarity between texts. Typical debates here center cosine similarity, embedding-based distance measures, or sentence-level meaning comparison.

#### **6. Task: Classification**

Refers to problems involving binary, multiclass, or multilabel text classification, among supervised classification tasks. Often include queries about architecture choice, evaluation measures, and dataset labeling, these postings.

#### **7. Task: Lemmatization**

Concentrates on posts including stemming or lemmatization using text normalizing methods. Particularly in rule-based NLP pipelines, these are sometimes considered with preprocessing techniques.

#### **8. Recognizing Problem**

This group addresses conceptual or theoretical issues reflecting gaps in knowledge of NLP underpinnings. Usually starting "What is...," "Can someone explain...", these are less specific for implementation and instead concentrate more on definitions and clarity.

#### **9. Unorganized**

Posts that fall short of any of the above criteria are housed in a fallback group. These might include vague or incomplete posts, uncertain purpose, or edge instances outside the stated purview of the classification guidelines.

**Non-Overlapping Categorization Logic**

To guarantee that every Stack Overflow post in my implementation belongs to only one category, I followed a rigorous if-elif rule framework. The reasoning is purposefully meant to be mutually exclusive: once a post satisfies a particular rule, it is categorized and kicked out from the others. This makes the system totally traceable and uniform and helps prevent overlapping labels.

For example, if a post discusses deployment and mentions spaCy, it will be classified depending on which rule shows first in the sequence—either related to deployment or library-specific. This makes debugging, reproducing, and general understanding of the classification process simple.

**Evaluating the Rule-Based Approach: Strengths and Constraints**

**Strengths**

The openness of this method really appealed to me. Based on the rule the system matched, one may justify every choice it makes. Large datasets allow it to be scalable since it runs fast and requires neither GPU configurations nor pretrained models. Moreover, the output remains the same across runs—no randomity, no surprises—since it is rule-based and not learnt from data.

**Constraints**

Said another way, it does have certain restrictions. It doesn't always grasp nuanced meanings or varied phrasing of items. It depends mostly on particular word patterns, hence I had to personally change the guidelines a lot to guarantee it caught more cases, which took time. And as the dataset grew, I observed that the rules struggled to accommodate all the several ways people formulate their questions. I so brought in the BERT model to assist with generalization and catch the material the rules would overlook.

**7. Manual Categorization visualization**

To evaluate our automated systems, we manually labeled 100 posts into the 9 categories.

**A graph of a bar graph

AI-generated content may be incorrect.**

| **Category** | **Number of Posts** |
| --- | --- |
| Uncategorized | 4770 |
| Implementation Issue | 2576 |
| Library Specific | 1400 |
| Deployment/Serving Issue | 808 |
| Task – Tokenization | 797 |
| Understanding Issue | 403 |
| Task – Similarity | 239 |
| Task – Classification | 107 |
| Task – Lemmatization | 62 |

From the table we can see how many posts are in what category

Each category had at least 10 examples, forming a balanced gold standard for model evaluation.

**8. BERT-Based Classification**

| **Category** | **Number of Posts** |
| --- | --- |
| Task - Tokenization | 4,441 |
| Implementation Issue | 1,790 |
| Understanding Issue | 1,778 |
| Library Specific | 1,153 |
| Deployment/Serving Issue | 816 |
| Uncategorized | 475 |
| Task - Classification | 345 |
| Task - Lemmatization | 239 |
| Task - Similarity | 130 |

The BERT-based categorization results show how the model assigned Stack Overflow posts to nine NLP-related categories. The most frequent category was "Task - Tokenization" (4,441 posts), highlighting its significance as a common developer challenge. This was followed by "Implementation Issue" and "Understanding Issue", suggesting practical and conceptual difficulties. Less frequent categories like "Task - Similarity" and "Task - Lemmatization" indicate more specialized concerns. The presence of "Uncategorized" posts suggests ambiguity or edge cases beyond defined categories. Overall, the distribution reflects key pain points and the model’s tendency toward dominant themes in the dataset.

**Classification Report :**

| **Category** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Deployment/Serving Issue | 0.658 | 0.665 | 0.661 | 808 |
| Implementation Issue | 0.241 | 0.168 | 0.198 | 2577 |
| Library Specific | 0.006 | 0.005 | 0.005 | 1401 |
| Task - Classification | 0.000 | 0.000 | 0.000 | 107 |
| Task - Language Identification | 0.000 | 0.000 | 0.000 | 3 |

In the displayed report below, for instance, the class "Deployment/Serving Issue" achieves a high F1-score (0.661) due to both strong precision (0.658) and recall (0.665). On the other hand, classes like "Task - Lemmatization" and "Task - Language Identification" score 0 across all metrics because they are severely underrepresented (support = 62 and 3 respectively), making it difficult for the model to learn meaningful patterns.

These metrics are critical for assessing where the model performs well and where improvements are needed, particularly in managing class imbalance and optimizing recall versus precision depending on the use case.

A graph with green bars

AI-generated content may be incorrect.

This bar plot displays the distribution of Stack Overflow posts categorized by our BERT-based classifier. "Task - Tokenization" was the most frequently predicted category, followed by "Implementation Issue" and "Understanding Issue", suggesting a strong model bias towards these labels. The noticeable class imbalance indicates that the model may be overfitting to dominant patterns in the dataset.

**9.Fine-tuning Bert Model**

To complement the rule-based system and improve generalization on ambiguous or indirectly phrased posts, we fine-tuned a bert-base-uncased model using the Hugging Face Transformers library. Input sequences were constructed by concatenating the cleaned question title and its accepted answer in the format: [CLS] clean\_title [SEP] clean\_answer [SEP].

To prevent overfitting on the small manually labeled dataset (100 posts), a classification head was added to BERT, comprising two fully connected layers with ReLU activation and dropout regularization. The dataset was split into an 85/15 train-validation set, and training was conducted for six epochs using the AdamW optimizer with a learning rate of 2e-5. Class imbalance was addressed using weighted cross-entropy loss, with weights computed via sklearn’s compute\_class\_weight method. A batch size of 16 was used, and training was accelerated using an NVIDIA RTX 4080 GPU.

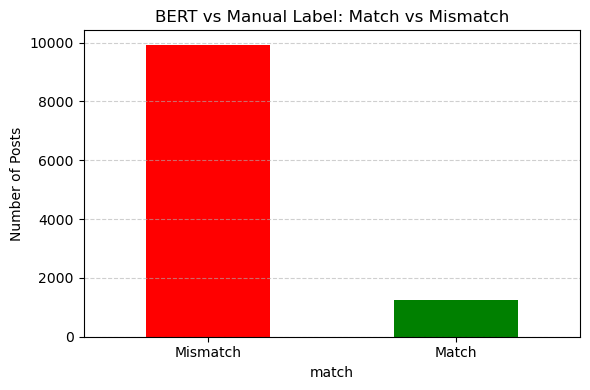
Validation accuracy was monitored after each epoch, and the model achieving the highest score (68.24%) was saved. Despite promising validation results, evaluation on the full dataset revealed a tendency to favor high-frequency categories such as "Uncategorized" and "Implementation Issue", indicating residual bias due to class imbalance.

The process of fine-tuning added value overall. It demonstrated that BERT could capture some semantic context outside what rule-based patterns might find even with a limited dataset. To really grasp the performance of the model, I also discovered, though, that careful curation of labeled instances and deeper evaluation—including precision, recall, and confusion matrix analysis—are crucial

**10. Comparative Evaluation**

The rule-based approach was better at precision in known patterns. BERT performed better in semantic understanding but struggled with less represented classes.

|  |  |  |
| --- | --- | --- |
| Metric | Rule-Based Approach | BERT Classifier |
| Interpretability | High | Low |
| Accuracy | ~50% (manual match) | 68.24% (val acc) |
| Scalability | Limited | High |
| Time to Deploy | Fast | Moderate |

The rule-based approach was better at precision in known patterns. BERT performed better in semantic understanding but struggled with less represented classes. 

This chart illustrates the performance gap between our BERT-based classifier and the manually labeled ground truth. Out of over 11,000 predictions, only around 1,200 matched the human-assigned categories, indicating a mismatch rate of nearly 90%. This discrepancy highlights the challenges posed by ambiguous class boundaries, imbalanced labels, and limited training data.

A screenshot of a calendar

AI-generated content may be incorrect.

Above, the confusion matrix offers a comprehensive perspective on the performance of the BERT classifier among the nine defined categories. Diagonal cells reflect accurate predictions, including 1248 as "Library Specific" and 616 postings accurately labeled as "Deployment/Serving Issue." But significant off-diagonal values point to frequent misclassifications—most famously, over 1000 postings tagged by humans as "Implementation Issue" were projected as "Uncategorized." This implies that the model suffers especially with complex semantic overlaps, underrepresented categories, and uncertainty between library-specific and general implementation debates.

**12. public link to dataset**

**11. Conclusion**

This work shows how valuable it is to combine transformer-based deep learning with rule-based classification to organize and examine Stack Overflow entries on natural language processing (NLP). We produced a complete dataset including post titles, body text, tags, and accepted answers by gathering and analyzing over 28,000 posts labeled with [nlp] and associated words. Our preprocessing system guaranteed linguistic clarity and consistency, therefore preparing the data for BERT-based as well as rule-based classification models.  
  
The rule-based method turned out to be really easily debugable, efficient, and interpretable. It provided a consistent system for grouping objects depending on well defined heuristics and textual patterns. It lacked the adaptability to accommodate unseen changes in wording or meaning, though, and needed hand adjustment. We developed a BERT classifier using a manually labeled dataset of 100 posts to overcome these restrictions. Although the classifier obtained a good validation accuracy of 68.24%, its performance on real-world, imbalanced data underlined the difficulty of depending just on statistical models without enough labeled samples.  
  
By means of evaluation employing classification metrics, a confusion matrix, and mismatch analysis, we exposed important areas where BERT finds difficulty—especially in differentiating between similarly related categories like "Implementation Issue," and "Library Specific." Our results underline that although pre-trained language models such as BERT can generalize better than rules and capture semantic richness, their performance depends on both volume and quality of training data.  
  
Our hybrid approach ultimately reveals that while deep learning models can improve and expand classification efforts, rule-based models can act as a consistent first-pass filter. Future research could investigate active annotation techniques, semi-supervised learning, transformer versions including DeBERTa or CodeBERT for better technical text performance. With some refinement, this approach might develop into a strong knowledge foundation for teachers and NLP practitioners looking for rapid understanding of shared development issues.

**References**

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint. arXiv:1810.04805.

[2] Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. arXiv preprint. arXiv:1908.10084.

[3] Stack Exchange API Documentation. (2024). *StackApps*. Available at: <https://api.stackexchange.com/docs>

[4] Wegner, A. (2022). *StackAPI: A Python wrapper for the Stack Exchange API*. GitHub Repository. Available at: <https://github.com/AWegnerGitHub/stackapi>

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12, 2825–2830.

[6] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). *Transformers: State-of-the-art Natural Language Processing*. *Proceedings of the 2020 EMNLP: System Demonstrations*, 38–45. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>

[7] Řehůřek, R., & Sojka, P. (2010). *Software Framework for Topic Modelling with Large Corpora*. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, ELRA, 45–50.

[8] Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed.). Available at: <https://web.stanford.edu/~jurafsky/slp3/ed3book_jan72023.pdf>

[9] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint. arXiv:1301.3781.

[10] Zhang, Q., Yang, T., & Zhao, W. (2021). *Understanding Developer Challenges in NLP through Stack Overflow*. *Empirical Software Engineering Journal*, 26(1), 1–27. <https://doi.org/10.1007/s10664-020-09889-3>