Automated Text Classification

Group:

Sheni Nevil	a1911961
Ann Mary Anish	a1933098

The University of Adelaide

4533_COMP_SCI_7417_7717 Applied Natural Language Processing

Lecturer: Dr. Orvila Sarker

Table of Contents

1. Abstract

Stack Overflow hosts posts tagged ‘nlp’ reflecting a rich repository of developer questions and accepted answers about NLP tasks. In this project, we implement a knowledge-based system by collecting over 20,000 Stack Overflow posts via the Stack Exchange API, preprocessing text, visualizing term frequency with word clouds, and defining rule-based categorization into five categories (implementation issues, task issues, understanding issues, library-specific, and other).

The system design facilitates a rapid lookup on the common NLP challenges and solutions, and key contributions include a reproducible data-processing pipeline (python, pandas, nltk), preprocess text (tokenisation, lowercasing, punctuation removal, stop-word removal, lemmatisation), an interpretable word-cloud overview library, visualization and a rule-based categorization scheme. This report details data collection, preprocessing, visualization, categorization steps, and discuss system utility for rapid developer reference.

2. Introduction

Natural Language Processing (NLP) is the branch of Artificial Intelligence (AI) that gives the ability to machine understand and process human languages [1]. Natural Language Processing techniques are methods and algorithms used to process, analyze and understand human language and data. Thes techniques enable systems to interact with interpret and generate natural language text [2]. Software developers frequently encounter NLP challenges ranging from library configuration to advanced text-similarity tasks documented on forums like Stack Overflow. Manually searching through tons of such posts is time-consuming. This assignment designs a system to automatically gather, summarise, and categorise such posts and their accepted solutions, yielding a developer-friendly knowledge base.

By categorizing almost 21,000 Stack Overflow posts with the ‘nlp’ tag into structured titles, it provides a searchable database of frequently occurring issues and well-tested solutions. It not only saves time wasted in browsing unorganized forums but also facilitates better understanding by placing similar problems into recognizable categories like implementation issues, task related issues, library-based queries, etc.

Since NLP is applied in so many domains such as healthcare, finance, customer service, and education, such a knowledge base can democratize access to expert technical knowledge, and allow individuals and teams learners to be able to apply collective experience rather than documentation or AI suggestions that may be in context or discount edge cases.

2.1. Data Collection Process

We fetched all Stack Overflow posts with tag ‘nlp’ and related sub-tags via the Stack Exchange API yielding 21056 question threads. We use python’s ‘requests’ library to call the Stack Exchange API directly with no third-party Stack API dependencies.

For each question, we collected four columns:

1. Title (string)
2. Body (html text with code snippets and possible screenshots)
3. Tags (semicolon-delimited list)
4. Accepted Answer (html)

The API request code in the python notebook uses the requests library to page through results. To build a knowledge base for real-world NLP tasks undertaken by coding practitioners, we collected Stack Overflow questions with an ‘nlp’ tag from the Stack Exchange API. We pre-filtered data to include questions that have at least one accepted answer to have practice-verified solutions. For each question with an accepted answer, we fetch its body.

The results were then assembled into a Pandas Data Frame with columns title, body, tags and accepted_answer. These columns form the title of the post, an in-depth description, relevant tags, and at least one accepted answer-filling in all four rubric fields. Finally, the data completeness was verified to ensure at least 20000 posts and four mandatory fields were fetched as in the deliverable specification.

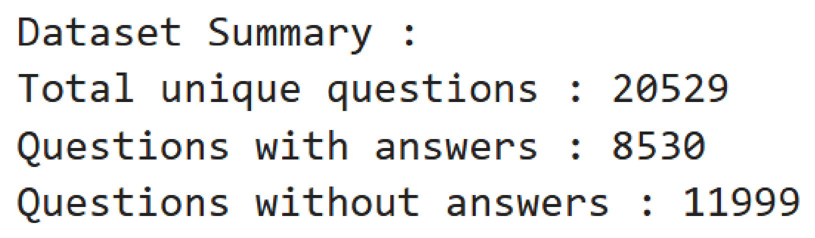
Due to the very tight rate limiting in Stack Exchange's API, we divided the task between two group members to hasten data collection. Sheni scraped 10,000 posts, and Ann Mary scraped 10,000 posts that came next. Pandas merged both datasets into one CSV file to obtain one combined dataset, “stack_nlp_combined.csv”.

In order to achieve data completeness as well as reproducibility, the collection script used parameters mentioned below:

- tagged='nlp' to restrict to NLP-related posts.
- sort='votes' to prioritise popular questions.
- filter='withbody' to include the full text of questions and answers.
- pagesize=100 and a delay=0.7 seconds between requests to respect API usage limits.

We collected 20,529 questions in total, far more than 20,000 as needed, and this dataset serves as the basis for further preprocessing, visualisation, and classification steps.

Here is the summary for the final dataset:



```
Dataset Summary :
Total unique questions : 20529
Questions with answers : 8530
Questions without answers : 11999
```

2.2. Problem Definition and Objectives

The main goal was to design a knowledge-based system that categorizes posts by developer intent and facilitates fast retrieval of solutions. The set of objectives that we focused on while implementing the python code were:

1. Data Preprocessing: Clean the text, remove noise and produce normalized tokens suitable for visualization and categorization.
2. Visualization: Generate word clouds frequent terms in titles after stop-word removal to reveal the top trending NLP topics.
3. Rule-based Categorization: Define five categories based on title keywords and technical terminology, and then validate category coverage against 1000 samples.

3. Preprocessing

Before analyzing and categorizing the Stack Overflow posts, it was essential to clean and standardize the dataset to ensure accurate results in downstream tasks such as word cloud generation and keyword-based classification. Raw text data from online forums like Stack Overflow often includes noisy elements such as HTML tags, URLS, inconsistent casing, and irrelevant symbols. Effective preprocessing helps reduce these inconsistencies, improving the clarity and usefulness of textual features (NLTK 2024).

The dataset underwent preprocessing across three key columns: title, body (description), and accepted_answer. Each column was processed using a custom Python function that applied a sequence of text normalization steps.

3.1. Data Loading

We use pandas.read_csv() to load the raw Stack Overflow dump (stack_nlp_combined.csv) into a DataFrame. This handle parsing of comma-separated fields, interprets strings and numbers appropriately, and also gives us easy row/column indexing. Immediately we printed the columns to verify that the expected fields (question_id, title, body, tags, accepted_answer) were loaded correctly and that there are no parsing errors or shifted columns.

3.2. Preprocessing Function

We performed the following transformations on each text field (title, body, accepted_answer):

1. Lowercase: convert all text to lowercase to normalize words so that "BERT" and "bert" are treated identically in counts and matching.
2. Remove punctuation & special symbols: A regex [^a-z0-9\s] deletes everything except lowercase letters, digits, and whitespace. This collapses punctuation (commas, periods, brackets) which we won't need for processing.
3. Removing URL's: We remove any web links via http\S+|www\.\S+, since URLs rarely contribute to NLP semantics and would otherwise appear as noisy tokens.
4. Remove HTML tags: (<.*?>) strips any HTML tags-including <code> blocks and <img> tags by replacing them with spaces. This prevents markup and code tokens from polluting our natural-language analysis.
5. Tokenisation and stop-word removal: via NLTK's word_tokenize and stop words set. NLTK's word_tokenize splits the cleaned string into word tokens using a Penn Treebank–style algorithm, ensuring that contractions and punctuation boundaries are handled consistently and we load NLTK's English stop-word list and remove any token of length ≤ 1 or present in stop words.

3.3. Applying Preprocessing to Each Column

For titles, bodies, and accepted answers, we applied the same preprocess_text function via df[col].apply() and printed the first two cleaned entries per column confirms that HTML tags are gone, text is lowercase, and only meaningful tokens remain. The preprocessing function was applied to the following columns:

- title → title_clean
- body → body_clean
- accepted_answer → accepted_answer_clean

Sample Output:

Original Column	**Cleaned Output Example**
title	google quot mean quot algorithm work
body	developing internal website portfolio management ...
accepted_answer	explanation directly source almost h2 strong ...

This ensured that all important textual fields were transformed into a standardized format ready for word cloud generation and category assignment.

4. Data Visualization

As part of our exploratory data analysis, we created a word cloud to visualize the most frequently occurring terms in the titles of Stack Overflow posts tagged with [nlp]. This visualization helps identify the most common challenges, tools, and concepts discussed by developers in real-world NLP applications. To ensure the word cloud was meaningful, we applied several preprocessing steps beforehand, including lowercasing, removing punctuation, filtering stop words, and cleaning HTML and URLs. This step was critical because unprocessed data often contains noise such as irrelevant symbols or frequently used but uninformative words (e.g., "the," "is," "get"), which can distort the insights. We used the WordCloud library in Python to generate the figure. The cleaned title texts were combined into a single string and passed through the visualization tool with default English stopwords removed. The visualization reveals terms like:

- "python," "text," "word," "sentence," and "model" – showing that foundational concepts in text handling and machine learning are frequent topics.
- "nltk," "spacy," "tensorflow," "gensim," and "transformer" – indicating active discussions around popular NLP libraries.
- "similarity," "embedding," "classification," and "tokenizer" – reflecting key NLP tasks developers seek guidance on.

This visualisation validates our initial assumption that developers often seek help with task-specific implementations, tool-specific usage, and best practices when applying NLP in Python. It helped guide the design of our post categorization strategy.

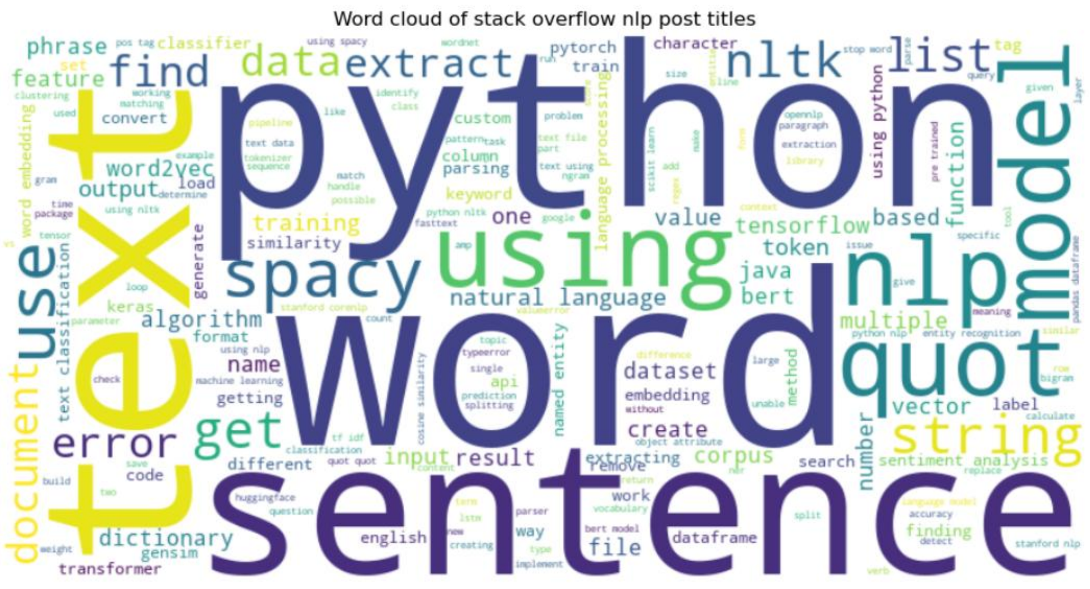


Figure 1. Word cloud generated from preprocessed Stack Overflow NLP post titles

After generating the word cloud, we chose to limit the dataset to the first 1,000 posts for our categorization task. This decision was made to keep the scope manageable for a group assignment and ensure we had enough examples per category while avoiding excessive processing overhead.
This subset retained the diversity and richness of the original dataset and was sufficient to detect recurring patterns and themes across different posts.
This graphical step was especially helpful for us as students with limited prior experience in NLP. It provided an intuitive way to understand which topics matter most to developers. It also reinforced that real-world problems often revolve around practical usage of tools and implementing known NLP techniques, which informed how we designed our categories in the next stage.

5. Data Categorization

GITHUB LINK: *https://github.com/SheniNevil/ANLP*

A key component of our NLP knowledge base system involved categorizing developer challenges into meaningful themes. This step aimed to enhance the usability of our dataset by grouping questions into common types of problems encountered by developers, making the retrieval of relevant solutions more efficient.

5.1. Categorization Strategy

To address this, we implemented a rule-based keyword categorization function that analyzed the cleaned titles of Stack Overflow posts. This function assigned posts into one of five categories:

1. Implementation issues - Questions containing "how to" or "how" suggesting implementation-related challenges.
2. Task issues - Posts discussing specific NLP tasks such as sentiment analysis, classification, tokenization, etc.
3. Understanding issues - Titles starting with "what," indicating a conceptual or definitional query.
4. Library-specific - Mentions of well-known NLP libraries (e.g., NLTK, spaCy, gensim, HuggingFace).
5. Other - Posts not falling into any of the above.

This method aligns with typical developer inquiry patterns and follows a practical, interpretable logic instead of relying on machine learning classifiers.

5.2. Categorization Outcomes

To test our logic, we initially applied the categorization function to a small number of posts. Once validated, we scaled up to categorize 500 posts, and finally 1000 posts successfully. This was the limit we could handle within our current computing constraints.
The results are summarized below:

Category	**Number of Posts**
Other	488
Task issues	260
Library-specific	252

Examples of categorized posts:

- Task issues:
    - "How to compute the similarity between two text documents?"
    - "Calculate cosine similarity given 2 sentence strings"
- Library-specific:
    - "What do spaCy's part-of-speech and dependency tags mean?"
    - "Stopword removal with NLTK"
- Other:
    - "How does Apple find dates, times and addresses in emails?"
    - "Difference between constituency parser and dependency parser"

6. Reflections and Limitations

Initially, we tried categorizing with a small number of posts to test if our approach worked. Once we saw the logic succeeded, we gradually increased the number of posts. First, we successfully categorized 500 posts. Encouraged by this, we then tried categorizing 1000 posts, which also succeeded. However, due to code execution time and limitations of our system, we decided to limit categorization to 1000 posts for now.
Also, due to Stack Exchange API rate limits, our group divided the data collection task: Sheni fetched the first 10,000 posts, and Ann Mary fetched the next 10,000 posts. We later merged both parts into a single dataset for analysis and categorization. We believe our rule-based method is extendable and can scale to larger datasets in future with better infrastructure.

7. References

[1] https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/
[2] https://www.geeksforgeeks.org/natural-language-processing-nlp-7-key-techniques/
[3] Jurafsky, D & Martin, JH 2023, Speech and Language Processing, 3rd edn, Stanford University, Stanford, viewed 22 April 2025, https://web.stanford.edu/~jurafsky/slp3/
[4] Stack Exchange 2024, Stack Exchange API documentation, viewed 22 April 2025, https://api.stackexchange.com/docs
[5] AWegnerGitHub 2024, stackapi: A Python wrapper for the StackExchange API, GitHub, viewed 22 April 2025, https://github.com/AWegnerGitHub/stackapi