Automated Text Classification

Shridharshan Veera Attur Krishnan	a1942516
Christen Infant Loyola	a1945001
Utsav Punia	a1956304

The University of Adelaide

4533_COMP_SCI_7417_7717 Applied Natural Language Processing

Lecturer: Dr. Orvila Sarker

Table of Contents

1. Abstract

This study involves the development of a rule-based system to categorize typical NLP-related problems raised by developers on Stack Overflow. Over 22,000 posts tagged with [nlp] and more were obtained from the Stack Exchange API, including titles, descriptions, tags, and accepted answers. The data underwent a structured preprocessing pipeline, after which it was partitioned for visualization and categorization tasks. A WordCloud was generated which emerged with prominently occurring NLP terms and sentence-level categorization was carried out using rule-based logic to combine titles and accepted answers. Classifications were given into problem categories such as Implementation Issues, Tool Usage, Text Preprocessing, and Sentiment Analysis. The system sets up a categorized knowledge base highlighting developer issues and solutions in NLP.

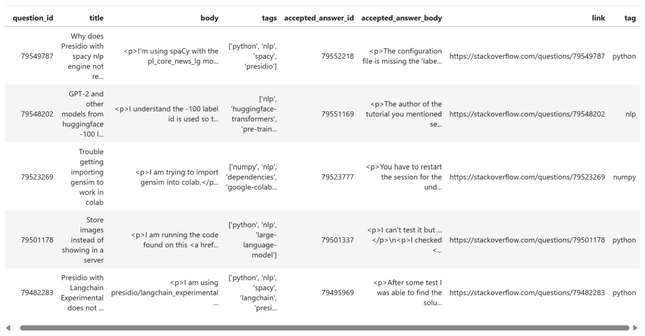
2. Introduction

Natural Language processing (NLP) plays a crucial role in modern software development, enabling machines to understand and generate human like language. Even though developers without prior experience in this field, they often face difficulties when performing basic tasks like text summarization, sentiment analysis, and sentence similarity. These challenges can be technical, conceptual, or tool related, and often lead developers to seek help from online communities like Stack Overflow. Forums like Stack Overflow provide real time insights into such challenges but the content is often unstructured. Our goal in this assignment is to structure these discussions by developing a knowledge base system aimed at supporting developers who are new to the field. Rather than relying solely on general purpose AI tools, we use community validated content from Stack Overflow to collect developer problems with their accepted answers. By collecting over 20,000 developer posts tagged with [nlp] or more from Stack Exchange API, the system identifies common issues, cleaning and preprocess them, and visualizing the common NLP terms, we categorize the posts using rule based techniques to create a structured reference of a real world developer experience that can be used for quick learning and decision making when performing NLP tasks and troubleshooting.

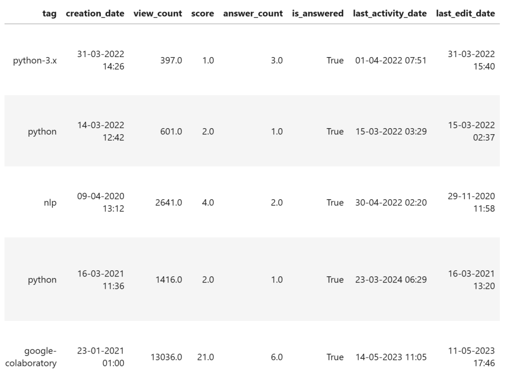
3. Data Collection

The first step of this assignment required us to acquire a dataset of over 20,000 Stack Overflow posts related to NLP. Given the scale of the task and the API limitations the process is devised strategically to ensure that the data acquired is of high quality and complete.*(Note:- External Sources like Stack Overflow and ChatGPT have been referred for assistance in designing the code for fetching the data.)*

The primary dataset extraction was done using the Stack Exchange API. The initial attempt was made without an API key, which limited the daily quota to about 300 requests per IP address. Despite trying batching and retry mechanisms, the request limit resulted in lost data and delays in the extraction process. We were able to successfully extract the fields required for the assignment which included question_id, title, body, tags, and accepted_answer_body, which were saved to a separate CSV file. (**Figure 1)**



question_id	title	body	tags	accepted_answer_id	accepted_answer_body	link	tag
79549787	Why does Presidio with spacy nlp engine not re...	<p>I'm using spaCy with the pl_core_news_lg mo...	['python', 'nlp', 'spacy', 'presidio']	79552218	<p>The configuration file is missing the 'labe...	https://stackoverflow.com/questions/79549787	python
79548202	GPT-2 and other models from huggingface -100 l...	<p>I understand the -100 label id is used so t...	['nlp', 'huggingface-transformers', 'pre-train...	79551169	<p>The author of the tutorial you mentioned se...	https://stackoverflow.com/questions/79548202	nlp
79523269	Trouble getting importing gensim to work in colab	<p>I am trying to import gensim into colab.</p...	['numpy', 'nlp', 'dependencies', 'google-colab...	79523777	<p>You have to restart the session for the und...	https://stackoverflow.com/questions/79523269	numpy
79501178	Store images instead of showing in a server	<p>I am running the code found on this <a href...	['python', 'nlp', 'large-language-model']	79501337	<p>I can't test it but ... </p>\n<p>I checked <...	https://stackoverflow.com/questions/79501178	python
79482283	Presidio with Langchain Experimental does not ...	<p>I am using presidio/langchain_experimental ...	['python', 'nlp', 'spacy', 'langchain', 'presi...	79495969	<p>After some test I was able to find the solu...	https://stackoverflow.com/questions/79482283	python



tag	creation_date	view_count	score	answer_count	is_answered	last_activity_date	last_edit_date
python-3.x	31-03-2022 14:26	397.0	1.0	3.0	True	01-04-2022 07:51	31-03-2022 15:40
python	14-03-2022 12:42	601.0	2.0	1.0	True	15-03-2022 03:29	15-03-2022 02:37
nlp	09-04-2020 13:12	2641.0	4.0	2.0	True	30-04-2022 02:20	29-11-2020 11:58
python	16-03-2021 11:36	1416.0	2.0	1.0	True	23-03-2024 06:29	16-03-2021 13:20
google-colaboratory	23-01-2021 01:00	13036.0	21.0	6.0	True	14-05-2023 11:05	11-05-2023 17:46

Figure 1. Dataset with 22,725 stack overflow posts **Figure. 2:** Dataset with 22,725 Stack Overflow Posts with 4 columns

This approach was enhanced by integrating an API key through the Stack Exchange API, increasing the quota to 10,000 requests per day. This adjustment allowed the team to collect the full dataset efficiently, adding additional information like creation_date, view_count, score, and answer_count to the dataset (**Figure 2)**, allowing them to understand more about the collected posts improving the dataset's analytical depth for further processing.

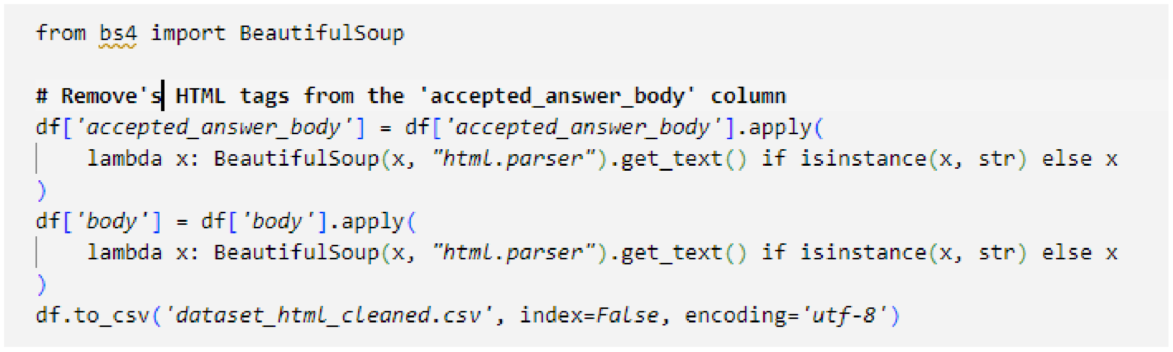
An alternate approach of using Stack Overflow's web-based query interface to filter and manually extract topic-relevant posts was also explored. While this method identified older questions more quickly, the lack of structured formatting and the inconsistency of the accepted answers limited an effective and reliable analysis. Due to these inconsistencies, this method was deemed unsuitable for scalable or reliable analysis and was not used for the final dataset.

Ultimately, combining all the structured information collected as a JSON file and then converted into a CSV file for easy processing, the final dataset(Figure 2) contains about 22,725 posts with four key columns and other important columns for further analysis.

4. Preprocessing

The preprocessing phase was a structured and collaborative effort aimed at preparing the Stack Overflow posts dataset for both categorization and visualization tasks. This process evolved through multiple iterations, where we not only refined forward but also revisited and revised earlier steps based on newly emerging requirements and observations.

4.1 Initial Pipeline Design



```
from bs4 import BeautifulSoup

# Remove's HTML tags from the 'accepted_answer_body' column
df['accepted_answer_body'] = df['accepted_answer_body'].apply(
    lambda x: BeautifulSoup(x, "html.parser").get_text() if isinstance(x, str) else x
)
df['body'] = df['body'].apply(
    lambda x: BeautifulSoup(x, "html.parser").get_text() if isinstance(x, str) else x
)
df.to_csv('dataset_html_cleaned.csv', index=False, encoding='utf-8')
```

Figure 3. Code to Remove HTML tags

We began with a basic preprocessing pipeline that ensures that a consistent format is maintained across all entries. This involves several foundational steps, such as removing HTML tags like <p>, <img>, and <code> (Figure 3) and converting all text to lowercase to maintain uniformity. Stripping URLs and special characters to minimize textual noise was also done, along with normalizing whitespace throughout the dataset. (Figure 3) These initial steps established a clean and standardized baseline, which ensured accuracy and reliability for all further processing tasks.

4.2 Dual-Track Preprocessing Strategy

After reviewing the assignment rubric and understanding the need for two different downstream tasks—rule-based categorization and word frequency visualization—we introduced a **dual-track strategy**, creating two tailored versions of the cleaned dataset:

Version	Purpose	Key Features
for_categorization	Used in rule-based categorization tasks	Retained stopwords, syntactic phrasing, and avoided lemmatization or stemming to preserve intent and sentence structure
for_visualization	Used in WordCloud and frequency analysis	Applied light stopword removal, excluded purely structural tokens, and preserved domain-specific terms like “bert”, “spacy”, and “token”

This separation allowed us to target specific goals—retaining grammatical cues for categorization, while enabling cleaner statistical representations for visualization.

4.3 Refinement and Rule Calibration

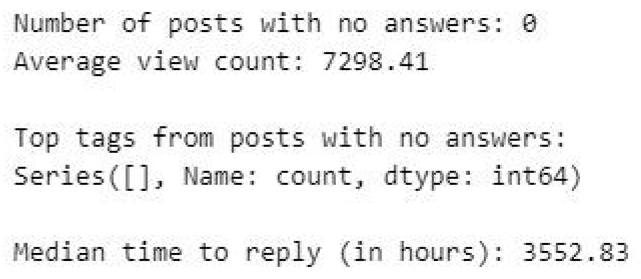
As the assignment progressed, we iteratively refined both versions of the dataset. In the categorization version, stopwords were deliberately preserved to retain the linguistic structure necessary for identifying question patterns such as "how do I" or "what is". Stemming and lemmatization were decided to be avoided to prevent distortion of user intent and phrasing. For the visualization version, stopword removal was applied more selectively, ensuring that NLP-related terms are retained to improve the quality of the word cloud. Additional refinements included handling missing values by removing rows with null entries in the accepted_answer_body column, as well as addressing encoding issues related to special characters and read failures during CSV parsing.

4.4 Revisiting and Iterating the Pipeline

As we moved ahead with categorization and began testing early rule patterns, we recognized limitations in our original cleaning strategy, especially where aggressive cleaning stripped away phrases critical for the classification task. In response, we revisited and revised our preprocessing pipeline, particularly for the categorization version, rolling back certain operations and reintegrating key structures like verbs and questions. This process of "going back in time" helped us adapt to the evolving goals of the assignment and ensured our final cleaned dataset was not only consistent but also fit-for-purpose across both technical and analytical components of the task. (Figure 2)

4.5 Metadata Recovery Post-Preprocessing

While working through preprocessing and revisiting the assignment rubric, we identified a specific requirement to visualize the dataset using metadata fields such as question scores, view counts, and answer counts. (Figure 4) These fields were not part of our initial data extraction.



```
Number of posts with no answers: 0
Average view count: 7298.41

Top tags from posts with no answers:
Series([], Name: count, dtype: int64)

Median time to reply (in hours): 3552.83
```

Figure 4. Average View Count and Median time to reply

Rather than re-fetching the entire dataset, which would have been inefficient and likely exceeded our API quota, we took a more optimized approach. We reacquired only the missing metadata by leveraging the question_id values already present in our preprocessed dataset. This allowed us to perform targeted API calls, significantly reducing data redundancy and saving time.

This adjustment demonstrates how we adapted our data pipeline dynamically based on rubric interpretation, and how we efficiently balanced API usage constraints with evolving assignment needs.

4.6. Final Preprocessing Function

A custom function was developed and applied across all relevant text fields. (Figure 3) It incorporated:

1. **Null Checks**: Skipped entries with missing values using pd.isnull().
2. **Lowercasing**: Normalized case for consistent matching.
3. **HTML Removal**: Removed <img>, <code>, and generic tags using re.sub().
4. **Link Removal**: Eliminated URLs beginning with http, https, or www.
5. **Punctuation & Symbol Stripping**: Retained only alphanumeric and whitespace characters using the pattern [^\w\s].
6. **Tokenization**: Applied word_tokenize() from NLTK for word-level token splitting.
7. **Stopword Removal**: Filtered using a standard English stopword list (for visualization-focused version).
8. **Reassembly**: Combined tokens into clean strings using '' .join (tokens).strip().

To prepare the dataset for reliable further analysis, two custom preprocessing functions (heavy_clean_text and light_clean_text) were implemented. These functions systematically applied standardized text normalization steps to ensure consistency, reduce noise, and optimize the dataset for two distinct tasks: visualization and categorization. Each step in the functions was chosen based on the characteristics of Stack Overflow posts and the rubric requirements of the assignment.

4.6.1 Handling Missing and Incomplete Data

The preprocessing functions begin by checking whether the input text is null or missing using pd.isnull(text). If an input is missing, an empty string is returned immediately. This step is essential to prevent the propagation of NaN values into downstream operations. Additionally, since the assignment required each post to have a valid accepted answer body, any records where the accepted_answer_body field was missing were excluded from the dataset prior to applying the final preprocessing. Posts without an accepted answer were considered incomplete and unsuitable for tasks like content categorization and summarization.

4.6.2 Heavy Cleaning for Visualization

The heavy_clean_text function performed extensive data cleaning, which prepared the dataset for analysis and generating the WordCloud visualization.[2]

Its main operations were:

- **Lowercasing**: All text was converted to lowercase using .lower() to enforce case insensitivity, reducing vocabulary fragmentation and improving matching accuracy during frequency-based analysis.

- **HTML Tag Removal**: HTML elements such as <img>, <p>, and <code> were stripped out using regular expressions, as they do not contribute semantic meaning and could skew NLP models or visualizations like word clouds.
- **URL Removal**: Hyperlinks starting with http, https, or www were removed to prevent non-informative tokens from distorting word frequencies.
- **Special Character Removal**: All non-alphanumeric characters (excluding whitespace) were removed using the pattern [^\w\s]. This helped standardize the text by eliminating punctuation and formatting artifacts.
- **Tokenization**: Cleaned text was tokenized into word-level units using word_tokenize() from NLTK, enabling structured word-level operations like stopword removal.
- **Selective Stopword Removal**: Common English stopwords were removed using NLTK's predefined list, except for a preserved set of domain-specific terms ("nlp", "bert", "spacy", "token", "huggingface", etc.). This enhanced the signal-to-noise ratio without losing important context.
- **Filtering Long and Numeric Tokens**: Tokens longer than 20 characters or purely numeric values were filtered out to eliminate noise.
- **Rejoining Tokens**: The final token list was reassembled into a clean string suitable for input into visualization tools (e.g., WordCloud, TF-IDF vectorizer).

4.6.3 Light Cleaning for Categorization

The light_clean_text (Figure 5) function was designed to perform minimal preprocessing for tasks that required maintaining sentence structure and grammatical clues important for rule-based categorization. Key differences included:

- Lowercasing the text to standardize casing.
- Removing only <img> tags and URLs, while retaining punctuation and phrasing.
- No stopword removal, no special character stripping, and no tokenization — preserving the text as close as possible to its original user-posted form.

This approach ensured that patterns like "how to", "can I", and "what is" remained detectable during manual or rule-based classification tasks.

4.6.4 Application Across the Dataset

Both functions were applied separately after deduplicating the dataset based on the title field:

- The **light-cleaned version** was saved as original_version.csv for manual reading and categorization.
- The **heavily cleaned version** was saved as preprocessed.csv for keyword visualization and further NLP tasks.

This structured two-version preprocessing pipeline allowed the project to satisfy both syntactic pattern-based categorization and keyword-driven visualization without conflict.

5. Data Visualisation

To visualize the selection of frequently used words by developers, we used a word cloud. In word cloud visualization, larger and bolder terms are highly frequent, whereas the smallest ones are less common terms More frequently used terms appear larger and bolder, whereas less common terms appear smaller. From the wordcloud visualization (Figure 6), we can see that words such as “python", "word", "sentence", "text," and "model " appear most frequently in the questions. This indicates that the developers often perform fundamental tasks such as text processing, model implementation, and Python-based tooling. [2]

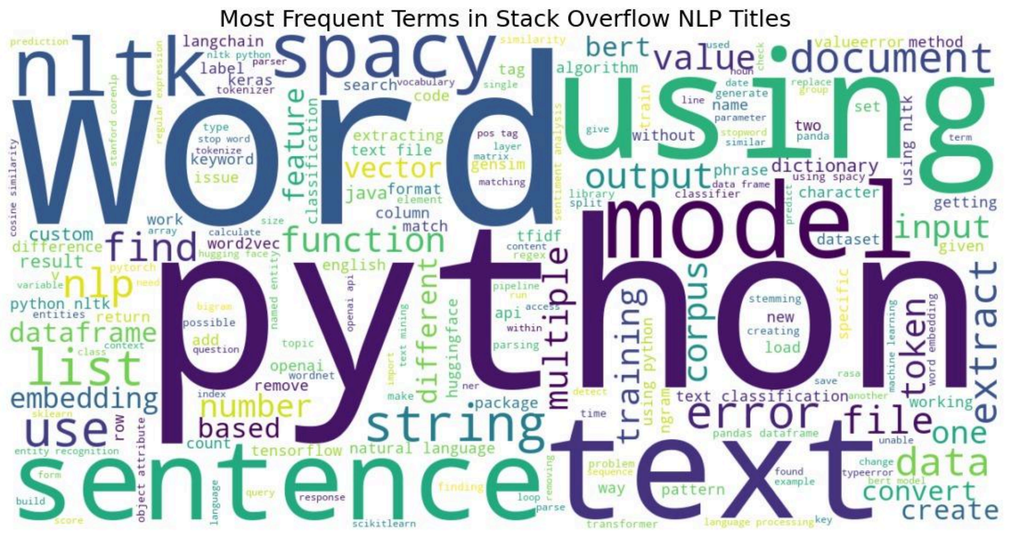


Figure 4. Wordcloud generated using most frequent terms in the posts.

6. Data Categorisation

To analyze and understand the trends from the Stack Overflow questions, all of the members used a rule-based, sentence-level categorization approach. This method was applied directly to the Stack Overflow dataset. Each team member used their own categorization techniques and methodologies for categorization.

Our method centered on defining logical rules to classify posts based on recognizable sentence patterns, such as questions beginning with “how to,” “what is,” or “difference between.” By emphasizing sentence structure over individual words, we aimed to enhance the accuracy and relevance of our classifications.

Each member categorized at least 100 posts, ensuring a minimum of 10 posts were assigned to each specified category. In the following subsections, each member's categorization methodology is briefly described.

6.1 Categorization of posts by Christen

Christen focused on implementing the core classification logic into a python function. The categorization strategy was developed through an initial manual review of about 10 to 20 stack overflow posts titles. Some common patterns were studied and these were translated into programmatic rules.n To begin with, the function categorize_post(), was designed to convert each post title to lowercase and then scan for key terms mapped to a predefined categories.

“Text Preprocessing” , triggered by terms like “tokenize”, "lemmatize", "remove stopword".

“Tool Usage” triggered by terms like "spacy", "nltk", "transformers".

“Model Training”triggered by terms such as "loss", "fit", "epoch".

“Conceptual Understanding” triggered by patterns like "what is", "explain", or "difference between".

“Other categories” included NER / Text Classification, Language Models, Sentiment Analysis, and Implementation Issues.

Category	Description
Text Preprocessing	Refers to questions about tokenization, punctuation removal, and text cleaning.
Model Training	Includes posts related to training models, tuning parameters, and validation.
Tool Usage	Covers issues or guidance around libraries like SpaCy, NLTK, or Transformers.
Implementation Issue	Represents posts describing technical errors, bugs, or installation problems.
NER / Text Classification	Captures posts focusing on named entity recognition or text classification.
Language Models	Involves posts referencing models like BERT, GPT, or other transformer-based LMs.

Other	Used for posts that don't clearly match any defined category.

A priority-based structure ensured that if multiple conditions were met, the most relevant category was assigned. If no matches were found, posts were labeled as Other. The function was applied across the dataset using pandas, and the output was saved in a new column for further evaluation.

6.2 Categorization of posts by Shridharshan

To categorize Stack Overflow posts related to NLP challenges, a sentence-level, rule-based function called improved_categorize_post() was created. The function analyzes both the post title and the accepted answer, merging them to capture the complete context of each question. The goal was to identify the intent, topic, or technical issue described in the post by applying a structured set of predefined rules.

Function Logic

- The title and accepted answer are combined into a single lowercase string for comprehensive analysis.
- This combined text is scanned for specific keywords and phrases related to common NLP tasks.
- A priority-based rule set ensures more task-specific categories are checked first, improving classification accuracy.

The categorization process includes the following categories:

Category	Description
Text Preprocessing Task	Posts involving tasks such as tokenization, stopword removal, lemmatization, and text cleaning.
Model Training Issue	Challenges related to model fitting, loss/accuracy, overfitting, epoch setup, or data splitting.
Language Detection	Queries about identifying the language of a given text or segment.

Sentiment Analysis	Posts discussing how to detect sentiment or opinion polarity (positive, negative, etc.).
NLP Task Query	Questions about performing common NLP tasks like summarization, classification, or NER.
Tool/Library Usage	Issues specifically tied to popular NLP tools like spaCy, NLTK, HuggingFace, BERT, or Word2Vec.
Implementation Issue	General implementation challenges expressed through patterns like “how to,” “not working,” or errors.
Conceptual Understanding	Posts seeking explanations of fundamental concepts, definitions, or comparisons in NLP.
Uncategorized	Posts that do not clearly match any of the above categories or fall outside predefined patterns.

6.3 Categorization of Posts by Utsav

We followed a two-phase rule-based logic:

1. **Phrase Matching**
   Posts were first scanned for common bigrams/trigrams that indicate question intent, such as “how to”, “what is”, “remove stopwords”.
2. **Keyword Scoring**
   If no phrase match was found, the post was scored by counting category-specific keywords. The category with the highest score was assigned.

This method ensures interpretability and replicates how a human would intuitively categorize a post based on sentence structure and context.

Final Categories and Detection Logic

Category Name	**Description**	**Detection Logic Keywords / Phrases**	**Example**
Implementation Issues	Posts asking how to implement code for an NLP task.	"how to", "how can I", "run code", "implement", "script"	*How to run sentiment analysis using NLTK?*
Preprocessing Tasks	Questions about cleaning, transforming, or preparing text data.	"remove stopwords", "clean text", "tokenize", "punctuation"	*Tokenize text and remove punctuation with regex*
Task-Specific Help	Questions on solving specific NLP problems like classification or similarity.	"classification", "topic modeling", "similarity", "NER"	*Cosine similarity for document clustering*
Theoretical Concepts	Questions asking for definitions, comparisons, or theory.	"what is", "difference between", "explain", "meaning of"	*What is the difference between stemming and lemmatization?*
Tool Setup / Errors	Errors or configuration issues related to NLP libraries/tools.	"install", "import error", "module not found", "path error"	*ImportError: No module named 'spacy'*
Data Wrangling	Queries about loading or formatting text data.	"read csv", "load data", "parse json", "extract text"	*How to load text data from a CSV file?*
Uncategorized	Posts that did not match any defined category rules.	—	—

Sentence-Level Compliance

The categorization function satisfies the sentence-level rule requirement outlined in the assignment. Each category is based on repeated phrasings that naturally occur in real-world developer queries, allowing the logic to simulate sentence-level understanding in a human-like, rule-driven form.

7. Conclusion

The project has led to the successful creation of a rule-based system intended to classify a variety of real-world problems posed by developers on the site Stack Overflow. The system has been classifying issues into classes such as "implementation errors," "tool-specific issues," and "core NLP task queries" by collecting structured information and preprocessing it before classifying sentence by sentence. The value of context on questions-and-answers substantially increases the efficiency of the classification. This has led to the transformation of unstructured talks into a searchable classed knowledge base to facilitate other developers with problem solution retrieval and scalable modalities for analyzing and researching real-life problems in NLP applications.

8. References

[1] Croft R., Xie Y., Zahedi M., Babar M. A., Treude C. 2021. An empirical study of developers' discussions about the security challenges of different programming languages. *Empirical Software Engineering* 27, 27. doi:10.1007/s10664-021-10054-w.

[2] Tahaei M., Vaniea K., Saphra N. 2020. Understanding privacy-related questions on Stack Overflow. *CHI'20: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* 639. doi:10.1145/3313831.3376768.

[3] ChatGPT Personal Communication (Data Collection, sentence structuring ,removing duplicates)

Link to GitHub Repository
https://github.com/Utsav1510/Assignment2_NLP