Assignment 2: Analysis of NLP Challenges on Stack Overflow

Nguyen Uyen Nhi Ong - a1906381

April 22, 2025

Abstract

This report gives a thorough examination of over 20,000 questions related to Natural Language Processing from website Stack Overflow. To gather the dataset, the Stack Exchange API was used to retrieve posts having the [nlp] tag and related ones. After implementing data cleaning and classification of these questions, we came up with key patterns and trends in the interests and obstacles are facing by developers in NLP areas. Our study includes information accumulation, preprocessing, visualization, and both rule-based and learning-based categorization approaches. The outcomes of learning-based predictor reveals three primary categorizations of NLP discussions: Python Scripting and Text Manipulation (accounts for the majority of questions at 71.3%), Deep Learning Architectures and Model Training (the next most prevalent at 22.0%), and Speech Recognition Applications and Conversational Agents (the smallest portion at 6.8%). We also observed a considerable change in the NLP field around 2015, when approaches realated to Deep Learning started getting more popular and widely used. This investigation gives better understanding into the development of the NLP field, which are reflected in the issues looked by developers all over the world.

1 Introduction

Natural Language Processing (NLP) combines linguistics, computer science, and artificial intelligence, it allows computers to analyze, interpret and generate human languages. As NLP becomes more and more common, it persists in every facet of our daily lives, from virtual assistants to translation services to recommendation systems, the NLP community has grown accordingly.

Stack Overflow, one of the largest Q&A platform for developer, is very likely to be a place where we can obtain the knowledge about the challenges, trends and advancements of NLP field. By analyzing NLP-related questions gathered from this platform, we can gain insights into:

- Common technical challenges faced by NLP practitioners
- Popular tools, libraries, and frameworks which are widely used in the community
- Shifts in methodology and approach over time
- Emerging topics and declining interests

This report explores a corpus of roughly 24,000 questions labeled with NLP and related tags from Stack Overflow accessed via the Stack Exchange API. From viewpoint of a developer, we seek to identify patterns that present in the past and present of the NLP field, through preprocessing, visualization, and classification data from Stack Overflow.

2 Methodology

2.1 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a probabilistic topic modeling technique, it is typically used to discover the hidden topics in a set of documents.

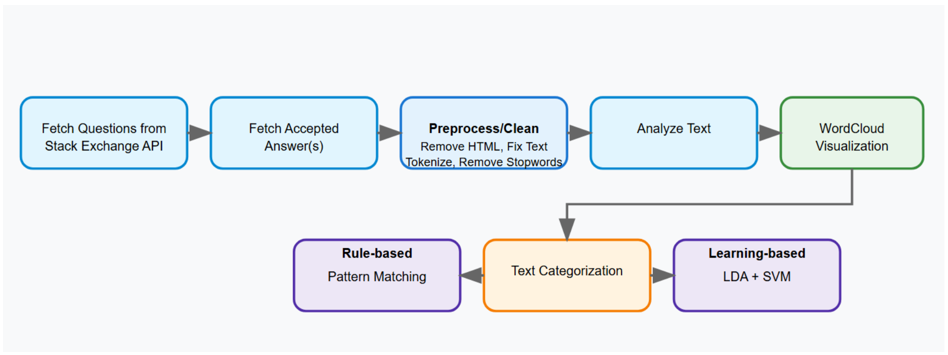

Figure 1: Workflow of Stack Overflow knowledge base system

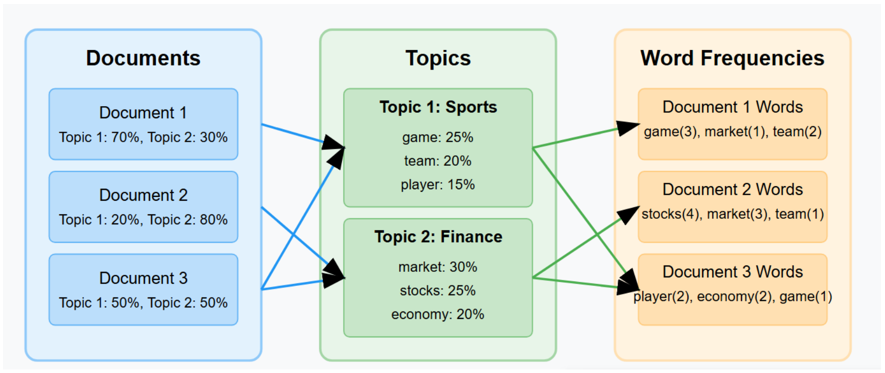

Figure 2: LDA Algorithm Visualization

Core concept: LDA assumes that documents are mixtures of topics, and topics are mixtures of words. The "latent" part refers to the hidden topic of a given document that the algorithm is trying to define.

How it works:

1. **Document Generation Process (in reverse)**: Each document has a discrete distribution of topics. Each topic has a distribution of words. Firstly, a document is generated by sampling a topic. Then we sample words for that document from the sampled topic (Figure 2)
2. **Dirichlet Distributions**: These are distributions of distributions, which are used as priors:
   - Document-topic distribution: How likely a topic appears in a document
   - Topic-word distribution: How likely a word appears in a topic
3. **Inference**: When given a set documents, LDA works backwards to find the most likely topic distributions that could have generated those documents.

LDA is unsupervised and it can reveal hidden patterns of given corpus that might not be recognizable to readers.

2.2 Linear Support Vector Classification (Linear SVC)

Linear Support Vector Classification (Linear SVC) is a powerful tool which is commonly used in classification tasks.

Core concept: Linear SVC tries to find the optimal hyperplane that maximizes the margin between different classes in our questions dataset, that hyperplane is also the best hyperplane to separate our data.

Mathematical Formulation

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \tag{2}$$

Decision function: $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

Geometric Interpretation (Figure 3)

Key Properties

- **Support Vectors**: Data points closest to the decision boundary
- **Margin**: Distance between hyperplane and support vectors
- **C parameter**: Controls trade-off between training error and margin

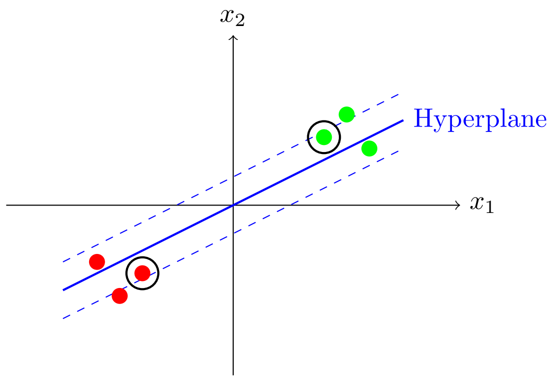

Figure 3: Linear SVC hyperplane with margin and support vectors

3 Data Collection and Insights

3.1 Data Source and Collection Process

Data for this analysis was collected from Stack Overflow using Stack Exchange API. We were trying to retrieve posts tagged with 'nlp' and related keywords, ensuring a comprehensive dataset relevant to Natural Language Processing. The following steps outline this process:

1. **API Setup**: The Stack Exchange API was accessed using an authenticated key to ensure higher rate limits and better access privileges (certain contents can only be accessible with an API key)
2. **Tags Selection**: The initial goal is to collect at least 20,000 NLP questions with accepted answers of the 'nlp' tag exclusively, however, we cannot meet that requirement. It led us to conduct a list of NLP-related tags for targeted data retrieval.
   - Primary tag: { 'nlp' }
   - Related tags: {'language-model', 'text-classification', 'word-embedding', 'spacy', 'nltk', 'seq2seq', 'sentence-similarity', 'named-entity-recognition' } and more
3. **Pagination Handling**: The API provider drew a line on the number of results that we can get per page, so we introduced pagination into the data retrieval process to retrieve all available data. A maximum of 30 questions were retrieved per page.
4. **Filter Settings**: The API requests we made specified parameters to retrieve only questions with accepted answers, sorted by creation date, and including body content.
5. **Rate Limiting**: To comply with the API's rate limits, a 1-second delay was added between requests to avoid being blocked by the API provider.

3.2 Data Description

The collected dataset consists of 24,774 unique Stack Overflow questions related to NLP, after removing duplicates. Each post in the dataset contains the following attributes:

Attribute	**Description**
question_id	Unique identifier for the question
title	Title of the Stack Overflow question
description	Body content of the question
tags	List of tags associated with the question
accepted answer 1	Content of the primary accepted answer
accepted answer 2	Content of the secondary answer (if available)
creation date	Unix timestamp of when the question was posted
view count	Number of times the question has been viewed
score	Score of the question

Table 1: Structure of dataset

3.2.1 Dataset Statistics

Key statistics from the dataset include:

Metric	Value
Total posts collected	29,638
Unique posts after deduplication	24,774
Average view count	3,218
Median view count	794
Average answer count	1.54
Posts with no accepted answers	0%

Table 2: Summary statistics of the collected dataset

Our dataset exceeds the minimum requirement of 20,000 posts which were labeled with [nlp] and related tags. We configured our data collection process to include only questions with accepted answers, which explains the 0% of posts with no answers.

3.2.2 Tag Distribution

Table 3 gives information about the distribution of tags of NLP posts, it revealed the most common related technologies and topics:

Tag	Frequency
python	12,169
nlp	8,508
deep-learning	3,406
nltk	3,308
tensorflow	2,736
machine-learning	2,685
keras	2,499
lstm	2,147
spacy	1,751
pytorch	1,506

Table 3: Top 10 most frequent tags in the dataset

This tag distribution highlights the dominance of Python, it is likely to be the most widely used programming language for NLP tasks, as well as the popularity of deep learning frameworks and specialized NLP libraries.

4 Data Preprocessing

Data preprocessing is a critical step to prepare data for the downstream NLP tasks. Next, we implemented a comprehensive preprocessing pipeline to clean and normalize text content of collected Stack Overflow posts.

In the first inspection, we found many instances in our dataset which had encoding problems, especially non-ASCII characters which carried no meanings. We decided to address this problem in the following manner:

- We wrote code to detect rows with encoding errors
- Applied `fix_text` method of `ftfy` library to correct encoding issues
- Used HTML unescaping to decode HTML entities
- Double-checked to make sure there is no longer non-ASCII characters in our corpus

4.1 Text Cleaning and Preprocessing

At least 4 main preprocessing techniques had been implemented to improve the quality of the dataset as required in the assignment rubric:

1. **Case Normalization:** Converted all text to lowercase to maintain consistency, ensuring words like "Python" and "python" are treated the same.

2. **Tokenization:** Used NLTK's `word_tokenize` function to split text into individual words, which is necessary for further analysis
3. **Stopword Removal:** Removed common English stopwords using NLTK's predefined list, because stopwords usually carry no distinguishable power.
4. **Lemmatization:** Applied spaCy to reduce words to their base forms (e.g., "running" → "run"), which is more sophisticated than stemming for technical text.

Along with the aforementioned preprocessing techniques, we also applied several different cleaning techniques:

- **Image Reference Removal:** Detected and removed texts referring to screenshots or images
- **Code Block Removal:** Eliminated inline code and code blocks inside HTML tags
- **HTML Tag Removal:** Removed all HTML tags while preserving the text content
- **URL Removal:** Filtered out web links that do not contribute to the textual content
- **Special Character Removal:** Eliminated punctuation and other non-alphanumeric characters
- **Number Removal:** Removed digits that typically do not carry semantic meaning in this context (programming discussions categorization)
- **Whitespace Normalization:** Normalized spacing between words

4.2 Preprocessing Results

The preprocessing pipeline significantly reduced noise and standardized text format in our data. Following is some key statistics: Titles are more concise with a small vocabulary, while question descriptions and

Column	Vocabulary Size	Avg. Tokens	Max Tokens
Title	11,340	5.99	20
Description	74,143	53.42	1,599
Tags	3,546	3.69	5
Accepted Answer 1	70,874	48.34	1,372
Accepted Answer 2	23,687	11.36	537

Table 4: Token statistics after applying preprocessing

answers contain richer and more diverse vocabulary sets. Tags have the smallest vocabulary set, which is expected.

5 Data Visualisation

Although standard libraries like NLTK provide comprehensive lists of English stopwords, these lists are insufficient when dealing with text from specialized domains. Looking at the cleaned text from previous stage, we saw that the common terms in programming discussions still persist, therefore, we conduct a custom technical stopwords list, in order to refine the vocabularies of each column, making them more NLP-oriented for WordCloud visualization. The custom stopwords list includes terms that are highly frequent in discussions about programming, however, they carry little distinguishable power as NLP terms, such as 'use', 'code', 'data', 'file', 'error', etc. We generated visual representations of the most common terms used in different Stack Overflow components in order to obtain better understanding of the content of our dataset. These graphs enable us to identify the main subjects and technologies covered in the NLP community.

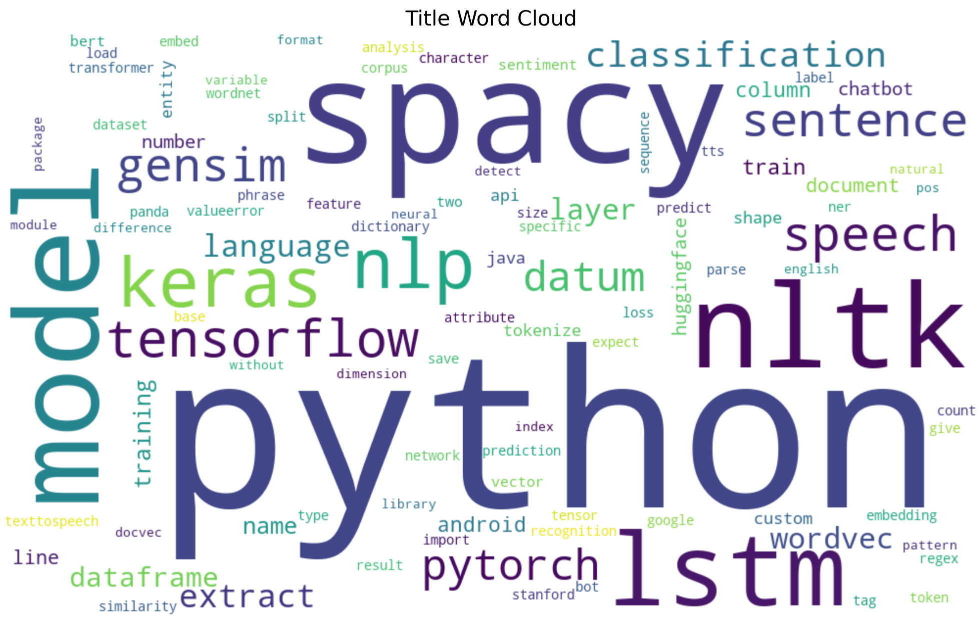

Figure 4: Word cloud of the most significant terms in post titles

5.1 Title Word Cloud

The word cloud generated from post titles provides insight into the main topics of interest in the NLP community:

We developed a sophisticated method for creating word clouds. Our aim was to make sure the word clouds included significant and unique NLP terms instead of noise words, as advised in the assignment criteria:

1. **Term Weighting with Multiple Factors:** Raw frequency, TF-IDF scores, bigram relevance, and domain relevance all combined
2. **Bigram Extraction:** Identified meaningful bigrams (e.g., "word embedding", "language model")
3. **Weight boosting based on tags:** Increased the weight of terms that also appear as popular tags
4. **Domain-specific weight boosting:** Increased the significance of important NLP terminology
5. **Custom Stopword Filtering:** Used custom stopword list created in previous step to filter out programming meta-language that would otherwise dominate the visualization

Advantages of this weighted word clouds strategy:

- Bigrams capture important technical technical that would be overlooked with single-word processing.
- By boosting terms based on tags, we can incorporate subject-matter expertise into the visualization.
- The combination of frequency, TF-IDF, and domain knowledge creates a more robust importance metric.

This approach ensured that our word clouds highlighted the most relevant and informative NLP-specific terms rather than generic programming terminology or common words. As a result, our visualizations provide clear insights into the actual NLP concepts and technologies being discussed, making the word clouds much more valuable for understanding the dataset content.

5.2 Key Insights from Title Word Cloud

Analysis of the title word cloud reveals several important patterns:

Dominant Technologies and Frameworks:

- **Python** is the most significant term, indicating the most used programming language for NLP tasks
- **spaCy** is also one of the largest terms, reflecting its popularity as an industrial-strength NLP library
- **LSTM** (Long Short-Term Memory) has significant prominence, showing the importance of this deep learning architecture

- **TensorFlow** and **PyTorch** both appear, representing the two most popular deep learning frameworks

Key NLP Concepts and Tasks:

- **Sentence** are prominent, representing fundamental units of text processing
- **Classification** is significant, suggesting many questions involving categorizing texts
- **Model** indicates a focus on machine learning approaches to solve NLP tasks
- **Tokenize** reflects the fundamental NLP preprocessing step
- **WordVec** (word vectors/embeddings) shows the importance of word representations
- **BERT** is visible which shows its impact on the field, though not as large as expected

Application Areas:

- **ChatBot** appears, showing interest in conversational AI
- **TextToSpeech** is visible, indicating interest in speech synthesis applications
- **Sentiment** suggests sentiment analysis is a common task
- **NER** (Named Entity Recognition) is present as a specific and common NLP task

6 Data Categorization

We introduced two approaches to categorize the Stack Overflow posts: a rule-based method operating at the word level using titles, and a learning-based approach that considered both titles and descriptions at the sentence level.

6.1 Rule-based Word-level Categorization

How rule-based approach works: Our first approach involved developing a rule-based system that categorizes posts based on regular expression patterns found in posts' titles. We first defined a set of regex expressions corresponding to different topics. Questions whose titles that match with at least one regex expression which belongs to one topic will be categorized into that topic.

We defined 3 main categories types and their matching regex expressions as follows:

1. **Question Types** - Based on the content of the question:
   - Implementation Questions: { "how to", "how do", "implement", "using" }
   - Conceptual Understanding: Identified by patterns like { "what is", "meaning", "explain", "understand" }
   - Troubleshooting: Identified by patterns like { "error", "issue", "problem", "not working" }
2. **NLP Tasks** - Based on the specific NLP function:
   - Tokenization & Preprocessing: { "tokeniz", "preprocess", "clean", "parse" }
   - Text Classification: Identified by patterns like { "classif", "categor", "sentiment", "predict" }
   - Named Entity Recognition: Identified by patterns like { "ner", "named entity", "extract entity" }
   - Word Embeddings: Identified by patterns like { "embedding", "word2vec", "glove", "vector" }
   - Language Models: { Identified by patterns like "language model", "bert", "gpt", "transformer" }
3. **Libraries & Technologies** - Based on technical tools:
   - spaCy Library: Identified by the pattern { "spacy" }
   - NLTK Library: Identified by patterns like { "nltk", "natural language toolkit" }
   - Hugging Face & Transformers: Identified by patterns like { "huggingface", "transformers" }

This approach allowed questions to be categorised into multiple categories at the same time. For example, a question could be an "Implementation Question" about "Named Entity Recognition" using the "spaCy Library". We then checked the match between configured regex patterns and posts' titles, the results are given as follows:

Category	**Number of Posts**
Implementation Questions	7,606
Troubleshooting	3,030
Text Classification	2,156
Tokenization & Preprocessing	1,723
NLTK Library	1,549
Speech Processing & Chatbots	3,190
spaCy Library	1,264
Word Embeddings	1,151
Named Entity Recognition	1,029
Language Models	1,030
Hugging Face & Transformers	551
Conceptual Understanding	711
Total unique posts categorized	15,480
Total unique posts in our dataset	24,774

Table 5: Rule-based categorization results

The category overlaps revealed strong relationships between certain categories:

- Implementation Questions + Text Classification: 736 posts
- Implementation Questions + Tokenization & Preprocessing: 578 posts
- Implementation Questions + NLTK Library: 530 posts
- Implementation Questions + spaCy Library: 424 posts
- Implementation Questions + Word Embeddings: 413 posts
- Implementation Questions + Language Models: 377 posts
- Hugging Face & Transformers + Language Models: 315 posts

We can see that ”Implementation Questions” frequently appears alongside with various NLP topics. The most common overlap is between ”Implementation Questions” and ”Text Classification”, which indicates that users are more likely to encounter technical difficulties or to have more questions about implementing text classification systems than other NLP tasks, it is also probably because there might be a gap between theoretical understanding of text classification and practical implementation. Other significant overlaps include ”Implementation Questions” paired with ”Tokenization & Preprocessing” (578 posts) and ”NLTK Library” (530 posts).

6.1.1 Strengths of rule-based approach

- **Explainability**: The patterns defined are relatively similar to human language, which makes the categorization process understandable
- **No training involved**: Unlike learning-based approaches, this rule-based implementation can be applied directly to test data without using any trained models
- **Multi-category classification**: This approach allows posts to be categorized into multiple categories at the same time
- **Domain-specific terminology driven**: The rules are tailored for NLP-related questions, capturing technical terminology relevant to the field

6.1.2 Weaknesses of rule-based approach

- **Limited flexibility**: The system only recognizes patterns that explicitly defined in the rules, potentially missing relevant posts that use different terminology. That is why there were only 15,480 posts categorized out of more than 24,000 posts in our dataset, a considerable amount of posts still stayed unlabeled.
- **No semantic understanding**: The approach relies on surface-level patterns rather than building knowledge about the actual meaning of the texts.
- **Sensitive to misspellings or abbreviations**: Words are sometimes misspelt and they also have their abbreviations, which can potentially lead to misclassification because predefined rules do not cover them.

6.2 Learning-based Sentence-level Categorization

For our second approach, we developed a hybrid method that combines unsupervised and supervised learning to categorize posts using both titles and descriptions of questions. **How learning-based approach works**: In this approach, we followed the following steps:

- **Use LDA to discover initial topics (unsupervised)**: We ran LDA models with predefined hyperparameter number of components from 2-6, which corresponds to 2-6 topics. Each model will return top keywords frequent inside each topics. We analyzed the resulted top keywords to come up with the most suitable number of topics for the following supervised learning model.
- **Review topics and refine them into meaningful categories**: based on the top keywords, renamed all categorizations
- **Training set creation**: sampled a small subset from original data (100 posts out of roughly 24,000), combined them with labels from selected LDA model to make it the train set
- **Train a classifier (SVM with TF-IDF features)**: calculated TF-IDF features for each document in the training set and used them as the inputs for SVM model
- **Apply the classifier to categorize all sentences in the dataset**

6.2.1 Topics Discovery with LDA

We applied Latent Dirichlet Allocation (LDA) to discover initial topics in the data by following these steps:

1. Created a document-term matrix using TF-IDF vectorizer
2. Experimented with different numbers of components (2-6 topics)
3. Analyzed the resulting ratio between topics and the interpretation of keywords for each topic corresponding to a specific number of components.

After evaluating different LDA models, we chose the hyperparameter number of components as 3, which resulted in a model that gave the most interpretable and balanced distribution:

- Topic 1 (6,104 posts): Keywords included ”model”, ”layer”, ”lstm”, ”train”, ”tensorflow”
- Topic 2 (15,480 posts): Keywords included ”word”, ”text”, ”file”, ”sentence”, ”python”
- Topic 3 (3,190 posts): Keywords included ”speech”, ”bot”, ”voice”, ”chatbot”, ”api”

Based on the dominant keywords mentioned above, we named the categories as follows:

1. Deep Learning & Model Training
2. Python Scripting & Text Handling
3. Speech Processing & Chatbots

6.2.2 Classifier Training using Linear SVC

Using the LDA results as initial labels, we went through these following steps to train SVC model and categorize questions in our dataset into 3 topics that we came up with from previous step.

1. Randomly sample a training set including 100 posts from original data (the other posts still stayed unseen to the SVC model)
2. Built a TF-IDF + LinearSVC classifier pipeline
3. Applied the trained classifier to all posts in the dataset

The classifier achieved an overall accuracy of 85% when compared to the LDA initial topic assignments, however, peformance of this classifier varies across categories:

- Deep Learning & Model Training: Precision 0.89, Recall 0.79
- Python Scripting & Text Handling: Precision 0.83, Recall 0.95
- Speech Processing & Chatbots: Precision 0.85, Recall 0.45

The final distribution of posts across categories was:

- Python Scripting & Text Handling: 17,699 posts (71.3%)
- Deep Learning & Model Training: 5,399 posts (22.0%)
- Speech Processing & Chatbots: 1,676 posts (6.8%)

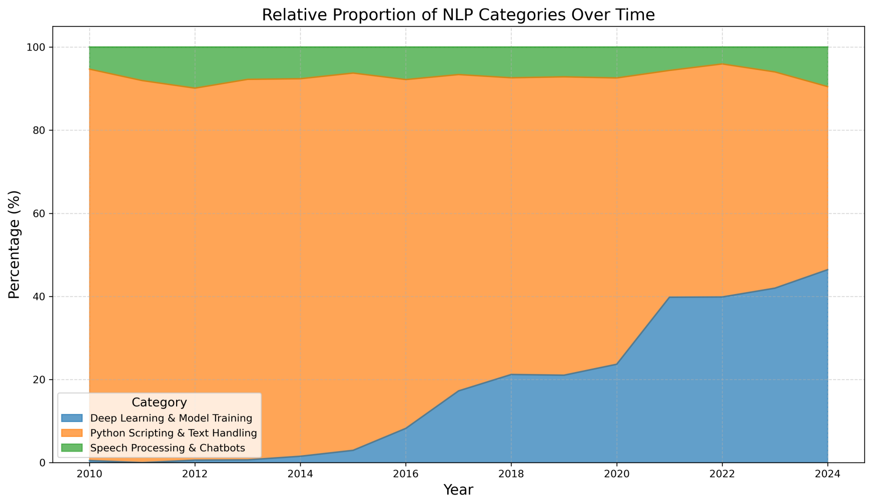

Figure 5: Relative Propotion of NLP Stack Overflow posts by category

Figure 5 illustrates the propotion of posts across three categories. The dominance of Python Scripting & Text Handling shows that basic text processing remains the foundation of NLP works, wheareas deep learning approaches have grown significantly in recent years.

6.2.3 Strengths of learning-based approach

- Scalability: Once trained on 100 posts, the model successfully classified approximately 24,000 posts across three categories.
- Semantic understanding: By using TF-IDF features as input data, SVC model can partly capture the semantic meaning inside the text, TF-IDF features are not raw frequency, it calculate the significance of a word inside a document, by that manner, similar documents contain similar sets of important words.
- Decent overall accuracy: The 85% of classification results are similar to LDA initial assignments suggests reasonable model consistency.

6.2.4 Weaknesses of learning-based approach

- **Imbalance between Categories**: The final distribution shows a significant imbalance with topic Python Scripting & Text Handling accounting for 71.3% of all posts
- **Low Recall for Speech Processing topic**: The low recall (0.45) indicates that many posts that belong to Speech Processing & Chatbots topic were misclassified.

7 Results

7.1 Code and Data Repository

All code, datasets, and detailed category information used in this project are available in our public GitHub repository at:

`https://github.com/KendrickLamawr/ANLP-A2/tree/product`

The repository includes:

- Complete dataset of 24,774 Stack Overflow posts with NLP and related tags
- Jupyter notebook includes scripts for data collection, preprocessing, and categorization (fully explained at each step)
- List of posts categorised by rule-based method
- List of posts categorised by learning-based method
- High-resolution versions of all visualizations

The code is thoroughly commented and includes clear documentation to make it understandable and executable by others.

7.2 Summary of Findings

This project successfully created a comprehensive knowledge base of NLP challenges and solutions based on Stack Overflow data. The key findings include:

- Python is the most common used programming language for NLP developments
- There are far more implementation questions than conceptual questions, highlighting Stack Overflow's role as a practical resource
- Text classification is the most discussed specific NLP task
- NLTK and spaCy are the most popular specialized NLP libraries
- Significant overlap exists between implementation questions and specific technical categories

This categorization scheme provides a useful framework for organizing NLP knowledge, allowing developers to quickly retrieve relevant information based on their specific needs.

7.3 Practical Applications

This knowledge base system has several practical applications:

- **Efficient Learning:** NLP learners can quickly find out the most common challenges and their solutions in specific NLP areas
- **Problem Anticipation:** developers can foresee possible risks or problems in their NLP projects by understanding the general patterns of recent discussions about NLP-related troubleshooting
- **Technology Selection:** When selecting a tech stack for NLP projects, it can be helpful to know which techniques are typically found in recent data.

7.4 Limitations

Several limitations should be considered when interpreting our findings:

- **Platform bias**: Stack Overflow might not represent all NLP works well because it reflects a specific community of practitioners, rather than those who work in the academic or industrial research areas, who are more likely to deal with theoretical and proof-related questions.
- **Decrease in questions frequency**: The decreasing trend in the number of NLP question after 2021 does not indicate a decline in NLP activities, it might be explained by the developments in documentation or shifts to other platforms like Hugging Face forums and GitHub discussions, it is necessary to combine data from other platforms beside Stack Overflow
- **General categories**: The categories provided in the analysis are broad and can be further refined for deep-diving analysis.
- **English-language focus**: The study focused on English posts only, which limits understanding of global NLP developments.

7.5 Future Work

Although this project provides valuable insights, there are still a lot of limitations which leave opportunities for future work

- **Temporal Analysis:** Studying how NLP challenges and technologies have evolved over time would provide additional insights
- **Answer Quality Analysis:** Further analysis of accepted answers would reflect common solution patterns and best practices (currently, we have just analyzed the titles and descriptions of those questions).
- **Questions' Difficulty Assessment:** It would enrich the knowledge base system by classifying questions in terms of complexity.
- **Sub-categorization:** break topics like "Implementation Questions" into parts smaller sub topics gives an opportunity for more thorough insights of the discussions involved.
- **Experiments on more advanced models**: Leveraging state-of-the-art transformer-based models like GPT-4 could significantly improve categorization accuracy. In the future, we will explore how these models handle ambiguous NLP questions and compare their performances against our current approaches.
- **Multi-source knowledge integration:** Expanding the dataset beyond Stack Overflow to include other technical forums would create a more comprehensive knowledge base.

8 Conclusions

This project demonstrates the significance of mining community knowledge from platforms like Stack Overflow to create structured knowledge base system. By categorizing and analyzing thousands of real-world NLP questions and their solutions, we have created a valuable reference that can be fascilitated to accelerate the learning progress of developers who are new to Natural Language Processing field.

References

[1] Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing (3rd ed.)*. Prentice Hall.

[2] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc. `https://www.nltk.org/`

[3] Explosion AI. *spaCy 3: Industrial-strength NLP*. `https://spacy.io/`

[4] The pandas development team. (2020). *pandas-dev/pandas: Pandas*. Zenodo. `https://doi.org/10.5281/zenodo.3509134`

[5] Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR, 12, 2825–2830. `https://scikit-learn.org/`

[6] Mueller, A. *wordcloud: A little word cloud generator in Python*. `https://github.com/amueller/word_cloud`

[7] Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95. `https://matplotlib.org/`

[8] Speer, R. (2015). *ftfy: Fixes Text For You*. `https://ftfy.readthedocs.io/`

[9] Stack Exchange Inc. *Stack Exchange API v2.3 Documentation*. `https://api.stackexchange.com/`

[10] OpenAI. (2024). *ChatGPT*. `https://openai.com/chatgpt`

[11] Google DeepMind. (2024). *Gemini*. `https://deepmind.google/technologies/gemini/`

[12] CoLearning Lounge. *NLP - Data Preprocessing and Cleaning*. Kaggle. `https://www.kaggle.com/code/colearninglounge/nlp-data-preprocessing-and-cleaning`

[13] Rajkumar, S. *Getting Started with Text Preprocessing*. Kaggle. `https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing`

[14] Islam, F. *NLP Tutorial 2: Text Preprocessing*. Kaggle. `https://www.kaggle.com/code/faizulislam19095/nlp-tutorial-2-text-preprocessing`

[15] Nguyen, L. *NLP Preprocessing & Feature Extraction Methods A-Z*. Kaggle. `https://www.kaggle.com/code/longtng/nlp-preprocessing-feature-extraction-methods-a-z`

[16] Kelde9. *Tutorial Preprocessing NLP — English & French Text*. Kaggle. `https://www.kaggle.com/code/kelde9/tutorial-preprocessing-nlp-english-french-text`

[17] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). *Latent Dirichlet Allocation*. JMLR, 3, 993–1022. `https://jmlr.org/papers/v3/blei03a.html`

[18] Griffiths, T. L., & Steyvers, M. (2004). *Finding scientific topics*. PNAS, 101(Suppl 1), 5228–5235.

[19] Řehůřek, R., & Sojka, P. (2010). *Software Framework for Topic Modelling with Large Corpora*. LREC. `https://radimrehurek.com/gensim`

[20] Cortes, C., & Vapnik, V. (1995). *Support-vector networks*. Machine Learning, 20, 273–297.

[21] Joachims, T. (1998). *Text categorization with Support Vector Machines*. ECML.

[22] Sokolova, M., & Lapalme, G. (2009). *A systematic analysis of performance measures for classification tasks.* Information Processing & Management, 45(4), 427–437.

[23] Mikolov, T., et al. (2013). *Efficient Estimation of Word Representations in Vector Space.* ICLR.

[24] Russell, M. A. (2013). *Mining the Social Web.* O'Reilly Media.

[25] Colah. (2014). *Visualizing and Understanding LDA.* `http://colah.github.io/posts/2014-10-Visualizing-LDA`

[26] Mueller, A. (2020). *Weighted Wordclouds in Python.* `https://amueller.github.io/word_cloud`

A Example of Rule-based Categorization Results

This appendix presents examples of categorized posts using rule-based approach

A.1 Conceptual Understanding

Question Title	**URL**
What is the difference between lemmatization and stemming?	`https://stackoverflow.com/q/1787110`
What exactly is Natural Language Processing?	`https://stackoverflow.com/q/4442627`
What's the difference between NER and POS tagging?	`https://stackoverflow.com/q/32763966`
What is the difference between word2vec and GloVe?	`https://stackoverflow.com/q/36302165`
What is the difference between supervised and unsupervised learning in NLP?	`https://stackoverflow.com/q/32010726`

A.2 Hugging Face & Transformers

Question Title	**URL**
How to properly use Hugging Face Transformers for sentiment analysis?	`https://stackoverflow.com/q/63600488`
Fine-tuning BERT for text classification with Hugging Face	`https://stackoverflow.com/q/63302171`
How to use Transformers from Hugging Face for multi-label classification?	`https://stackoverflow.com/q/65112354`
How to use Hugging Face's pipeline for zero-shot classification?	`https://stackoverflow.com/q/67670591`
What are the differences between Transformer models available in Hugging Face?	`https://stackoverflow.com/q/62344510`

A.3 Implementation Questions

Question Title	**URL**
How to implement Word2Vec model with Gensim?	`https://stackoverflow.com/q/47064069`
How to implement TF-IDF from scratch in Python?	`https://stackoverflow.com/q/35562074`
How to implement text summarization with Python?	`https://stackoverflow.com/q/34232190`

How to implement a custom NER model with spaCy?	`https://stackoverflow.com/q/44827930`
How to implement topic modeling with LDA in Python?	`https://stackoverflow.com/q/20349958`

A.4 Language Models

Question Title	**URL**
How to create a simple language model with LSTM in Keras?	`https://stackoverflow.com/q/48071236`
What is the difference between BERT and GPT language models?	`https://stackoverflow.com/q/62730706`
How to use GPT-2 for text generation in Python?	`https://stackoverflow.com/q/60034655`
Training a custom language model with transformers library	`https://stackoverflow.com/q/64240924`
How to evaluate language model performance with perplexity?	`https://stackoverflow.com/q/54941121`

A.5 Named Entity Recognition

Question Title	**URL**
How to extract named entities using NLTK in Python?	`https://stackoverflow.com/q/31836058`
Training a custom NER model with spaCy	`https://stackoverflow.com/q/44827930`
How to extract person names from text?	`https://stackoverflow.com/q/20290870`
Comparing NER accuracy across different libraries	`https://stackoverflow.com/q/55175294`
How to recognize custom entities with BERT-based NER?	`https://stackoverflow.com/q/62509459`

A.6 spaCy Library

Question Title	**URL**
How to extract noun phrases with spaCy?	`https://stackoverflow.com/q/48959098`
How to add custom stop words in spaCy?	`https://stackoverflow.com/q/41170726`
Training custom NER model with spaCy 3.0	`https://stackoverflow.com/q/65879018`
How to use rule-based matching in spaCy?	`https://stackoverflow.com/q/40288323`
How to calculate text similarity with spaCy?	`https://stackoverflow.com/q/55921104`

A.7 Text Classification

Question Title	URL
How to implement multi-class text classification with scikit-learn?	`https://stackoverflow.com/q/45732838`
Best approach for sentiment analysis in Python	`https://stackoverflow.com/q/10135684`
How to implement BERT for text classification?	`https://stackoverflow.com/q/56771260`
Creating a custom text classifier with TF-IDF and SVM	`https://stackoverflow.com/q/34149580`
How to improve accuracy of text classification model?	`https://stackoverflow.com/q/57668671`

A.8 Tokenization & Preprocessing

Question Title	URL
How to tokenize text in Python?	`https://stackoverflow.com/q/9001509`
How to remove stopwords with NLTK?	`https://stackoverflow.com/q/19130512`
Stemming vs. Lemmatization - which to use?	`https://stackoverflow.com/q/17317418`
How to handle contractions in text preprocessing?	`https://stackoverflow.com/q/19790188`
Efficient way to preprocess large text corpus	`https://stackoverflow.com/q/39782418`

A.9 Troubleshooting

Question Title	URL
Error in getting Captum text explanations for text classification	`https://stackoverflow.com/q/79247672`
SeqSeq Trainer.train keeps giving indexing error	`https://stackoverflow.com/q/79005985`
Trainer Hugging Face RuntimeError: cannot pin torch.cuda.FloatTensor	`https://stackoverflow.com/q/78949607`
Finetuning a pretrained model with quantization and amp scaler error	`https://stackoverflow.com/q/78943401`
spaCy Matcher with optional suffix in pattern reports multiple matches	`https://stackoverflow.com/q/78865486`

A.10 Word Embeddings

Question Title	URL
How to use pre-trained word2vec models?	`https://stackoverflow.com/q/32545181`
How to visualize word embeddings using t-SNE?	`https://stackoverflow.com/q/43776572`
Word2Vec vs. GloVe - which one to use?	`https://stackoverflow.com/q/38568776`

How to create custom embeddings for domain-specific vocabulary?	https://stackoverflow.com/q/51636612
Using word embeddings for document similarity	https://stackoverflow.com/q/42781292

B Example of Learning-based Categorization Results

This appendix presents examples of categorized posts using learning-based approach

B.1 Deep Learning & Model Training

Question Title	**URL**
Trouble getting importing gensim to work in colab	https://stackoverflow.com/q/79523269
Underfitting pretrained GloVe LSTM model accurcacy unchanged	https://stackoverflow.com/q/79419884
How to get custom column in the models forward function when training with Huggingface Trainer	https://stackoverflow.com/q/79328514
Normalization of token embeddings in BERT encoder blocks	https://stackoverflow.com/q/79178041
Is it possible to get embeddings from nvembed using candle	https://stackoverflow.com/q/79145419

B.2 Python Scripting & Text Handling

Question Title	**URL**
Store images instead of showing in a server	https://stackoverflow.com/q/79501178
Presidio with langchain experimental does not detect Polish names	https://stackoverflow.com/q/79482283
OpenNLP POSTaggerME and ChunkerME synergy	https://stackoverflow.com/q/79459888
Word sentence similarities	https://stackoverflow.com/q/79451974
Can't compile Marian NMT	https://stackoverflow.com/q/79330283

B.3 Speech Processing & Chatbots

Question Title	**URL**
Cannot import name 'split_torch_state_dict_into_shards' from 'huggingface_hub'	https://stackoverflow.com/q/78920095
How can I use structured_output with Azure OpenAI with the OpenAI Python library	https://stackoverflow.com/q/78846004
How to make dynamic API calls based on user input in a Gemini application Python NLP	https://stackoverflow.com/q/78707861
Not able to install spacy version	https://stackoverflow.com/q/78538749
Langchain SQL agent with context	https://stackoverflow.com/q/78394225