Final Project Report

Neena Chaudhari

2023-12-05

Title

Tag Prediction on Stack Exchange

Team Members

Shravan Honade

Ritesh Sengar

Nikhil Patil

Dhananjay Ghate

Neena Chaudhari

Problem Description

This project aims to use data and modeling to explore and predict tags for questions on Stack Exchange by using Natural Language Processing (NLP) with Keras Word2Vec and Paragraph2Vec technique.

1. **Data Extraction**: The initial step of the project focuses on extracting relevant data from the Stack Exchange API, which includes question texts, associated tags, and various metadata.
2. **Exploratory Data Analysis (EDA)**: After data extraction, a thorough EDA will be conducted to gain insights into the dataset. This analysis will encompass data distribution, question-answer analysis length, word frequency, and other exploratory aspects.
3. **Machine Learning Model Development**: The core of the project involves developing a machine learning model that can predict the most appropriate tag for a given question based on the historical data. The model may utilize natural language processing (NLP) techniques, feature engineering, and classification algorithms to achieve this task.

The project aims to improve the efficiency of question tagging on Stack Exchange by automating the process with a reliable machine learning model. This will enhance the user experience, reduce manual effort, and promote a better organization of the platform's content."

Dataset URL

https://drive.google.com/file/d/1ObWNw-qRmg1ZjzDt5v2V2h1r6JR3PiYh/view?usp=sharing (https://drive.google.com/file/d/1ObWNw-qRmg1ZjzDt5v2V2h1r6JR3PiYh/view?usp=sharing)

Data summary, exploration, and discussion

- Data Retrieval using Stack Exchange API in R.
- Number of Rows - 650K+ and Columns - 11
- Summary of columns -
  - tags: Categories or labels associated with the questions.
  - question_id: Unique identifier for each question.
  - title: Title of the question.
  - is_answered: Indicates whether the question has been answered (TRUE or FALSE).
  - view_count: Number of views the question has received.
  - answer_count: Number of answers the question has.
  - score: Score assigned to the question based on votes.
  - last_activity_date: Timestamp of the last activity on the question.
  - creation_date: Timestamp of when the question was created.
  - last_edit_date: Timestamp of the last edit made to the question.
  - content_license: License under which the content is distributed.
  - link: URL link to the question on Stack Overflow.

Data

```
example_data <- head(df, 5)
knitr::kable(
  example_data,
  caption = "Stack Exchange Dataset",
  format = "html",
  full_width = FALSE
)
```

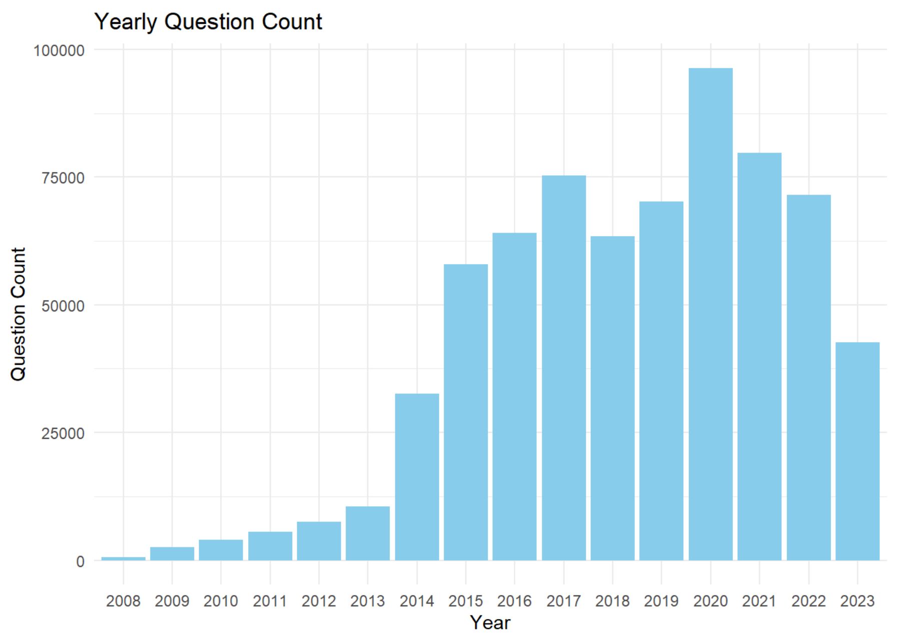
Stack Exchange Dataset

tags	question_id	title	is_answered	view_count	answer_count	score	last_activity_date	creation_date	last_edit_date	content_lic
python|contour|scipy-optimize|data-fitting|uncertainty	77592064	How do I find the uncertainties on a fit with correlated parameters in python?	False	4	0	0	1701548773	1701548773	NA	CC BY-SA 4
python	77591142	How to check whether a string is Palindrome or not in Python programme?	True	57	3	-3	1701548575	1701533698	NA	
python|python-import|relative-path|relative-import	73991299	Importing a Python module from subfolder of another folder using relative path	True	1003	2	1	1701547786	1665167181	NA	CC BY-SA 4
python|gitlab|repository	77547154	Python Code to Check Multiple Files on Gitlab repository by group ID	False	30	1	1	1701547456	1700900779	NA	CC BY-SA 4
python|image-processing|python-imaging-library|huffman-code	77591994	Huffman Encoding/Decoding - Black Image Issue. [Python]	False	9	0	0	1701547358	1701547358	NA	CC BY-SA 4

Exploratory Data Analysis

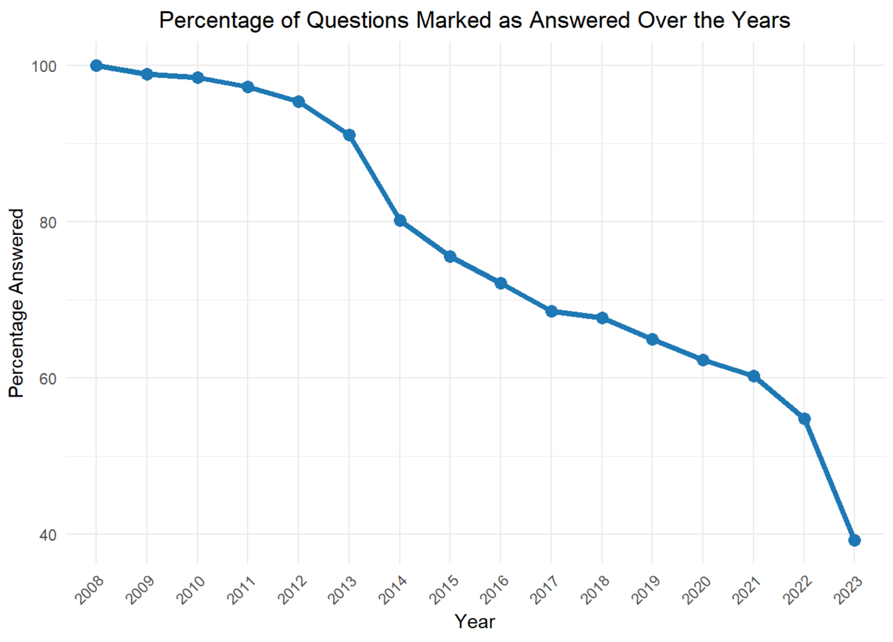
1. Yearly questions count

```
df_tags_binary_monthly_count <- df %>%
  mutate(creation_date = as.POSIXct(creation_date, origin = "1970-01-01", tz = "UTC")) %>%
  mutate(year = format(creation_date, "%Y"))

# Create a bar chart
ggplot(df_tags_binary_monthly_count, aes(x = year)) +
  geom_bar(stat = "count", fill = "skyblue") +
  labs(title = "Yearly Question Count", x = "Year", y = "Question Count") +
  theme_minimal()
```



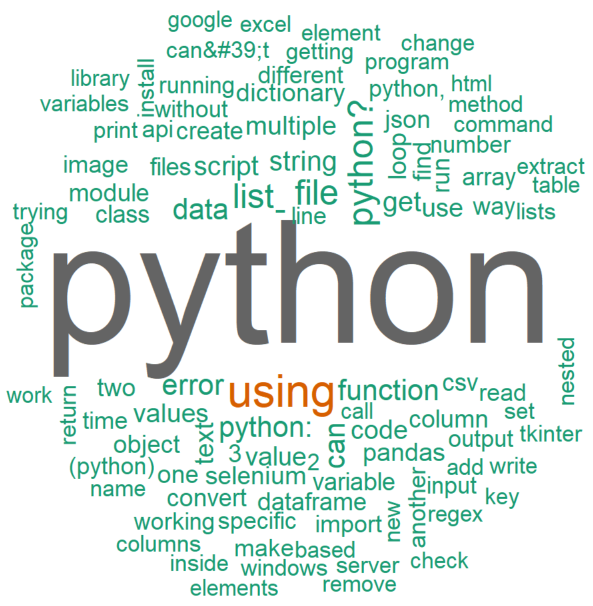
2. Percentage marked as answered vs. questions

```r
# Assuming 'last_activity_date' and 'creation_date' are in Unix timestamp format
df$last_activity_date <- as.POSIXct(df$last_activity_date, origin = "1970-01-01", tz = "UTC")
df$creation_date <- as.POSIXct(df$creation_date, origin = "1970-01-01", tz = "UTC")

# Extract the year from the date columns
df$year <- format(df$creation_date, "%Y")

# Convert 'is_answered' to logical (if not already)
df$is_answered <- as.logical(df$is_answered)

# Calculate the percentage of questions marked as answered for each year
answered_percentage <- df %>%
  group_by(year) %>%
  summarise(percentage_answered = mean(is_answered, na.rm = TRUE) * 100)

# Define custom color palette
custom_palette <- c("#1f78b4", "#33a02c")  # Blue and green

# Create a beautiful plot
ggplot(answered_percentage, aes(x = year, y = percentage_answered, group = 1)) +
  geom_line(color = custom_palette[1], size = 1.5) +
  geom_point(color = custom_palette[1], size = 3) +
  labs(title = "Percentage of Questions Marked as Answered Over the Years",
       x = "Year",
       y = "Percentage Answered") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_color_manual(values = custom_palette)
```



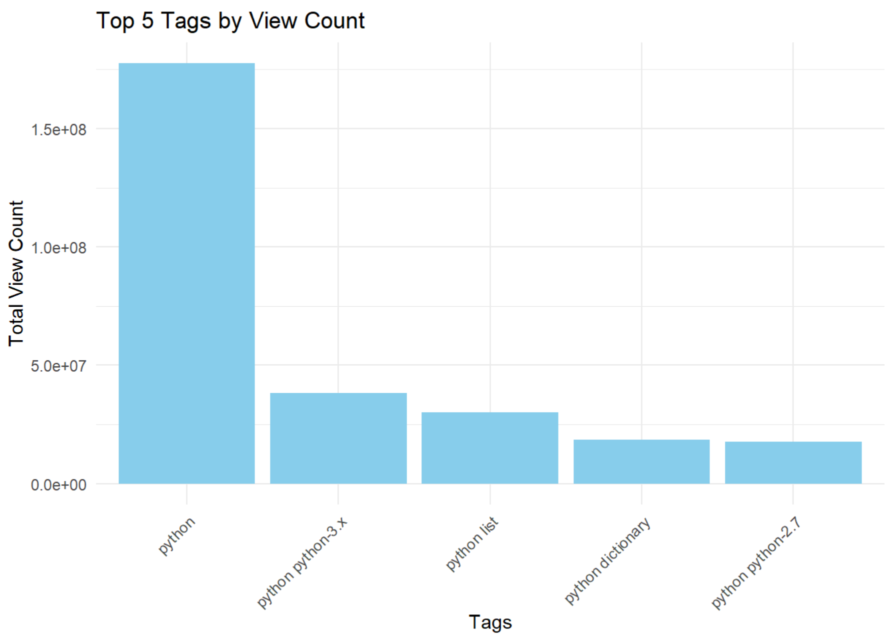
3. Common words/tags in titles

```
# Combine all titles into a single character vector
all_titles <- paste(df$title, collapse = " ")

# Convert to lowercase to ensure case-insensitive counting
all_titles <- tolower(all_titles)

# Tokenize the text into words
title_words <- unlist(strsplit(all_titles, "\\s+"))

# Filter out common English stop words if needed
title_words <- title_words[!title_words %in% stopwords("en")]

# Create a table of word frequencies
word_freq <- table(title_words)

# Convert the table to a data frame
word_freq_df <- as.data.frame(word_freq, stringsAsFactors = FALSE)
colnames(word_freq_df) <- c("word", "freq")

# Sort by frequency in descending order
word_freq_df <- word_freq_df[order(-word_freq_df$freq), ]

# Limit the number of words in the word cloud
num_words_to_plot <- 100
word_freq_df <- head(word_freq_df, num_words_to_plot)

# Plotting the word cloud
wordcloud(words = word_freq_df$word, freq = word_freq_df$freq, scale = c(8, 1), min.freq = 1, colors = brewer.pal(8, "Dark
2"), random.order = FALSE)
```



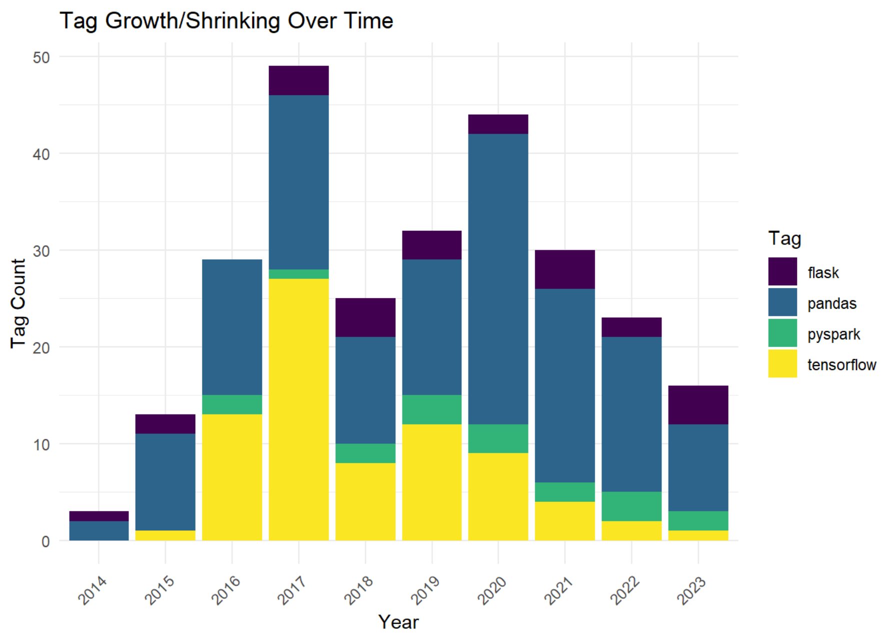
4. Top 5 tags by Total View Count

```
# Assuming df is your data frame
# Convert view_count to numeric (in case it's not already)
df$view_count <- as.numeric(df$view_count)

# Create a table of tag frequencies
tag_view_counts <- df %>%
  group_by(tags) %>%
  summarise(total_views = sum(view_count)) %>%
  arrange(desc(total_views)) %>%
  head(5)

# Create a bar plot of the top 5 tags by view count
ggplot(tag_view_counts, aes(x = reorder(tags, -total_views), y = total_views)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 5 Tags by View Count", x = "Tags", y = "Total View Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability
```



5. Tag Growth/Shinking for Top 5 tags

```
# Extract relevant columns and convert creation_date to POSIXct
df_tags_time <- df %>%
  select(tags, creation_date) %>%
  mutate(creation_date = as.POSIXct(creation_date, origin = "1970-01-01", tz = "UTC"))

# Extract year from creation_date
df_tags_time <- df_tags_time %>%
  mutate(year = format(creation_date, "%Y"))

# Create a table of tag frequencies over time
tag_counts_over_time <- df_tags_time %>%
  separate_rows(tags, sep = "\\|") %>%
  count(year, tags)

# Filter for the top 5 tags
top_tags <- tag_counts_over_time %>%
  group_by(tags) %>%
  summarise(total_count = sum(n)) %>%
  arrange(desc(total_count)) %>%
  head(5)

manual_tags <- c("pandas", "tensorflow", "pyspark", "flask")

# Filter the main data frame for the specified tags and a specific range of years (e.g., 2020 to 2022)
df_manual_tags_filtered <- df %>%
  filter(str_detect(tags, paste(manual_tags, collapse = "|"))) %>%
  filter(creation_date >= as.POSIXct("2020-01-01", tz = "UTC") & creation_date <= as.POSIXct("2022-12-31", tz = "UTC"))

# Plotting tag growth/shrinking over time using a bar chart
ggplot(tag_counts_over_time %>% filter(tags %in% manual_tags), aes(x = year, y = n, fill = tags)) +
  geom_col(position = "stack") +
  scale_fill_viridis_d() +  # You can choose a different color scale if needed
  labs(title = "Tag Growth/Shrinking Over Time", x = "Year", y = "Tag Count", fill = "Tag") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability
```



Keras Word2Vec Model

Word2Vec is a tool that learns compact numerical representations (vectors) for words by analyzing how they are used together in a given set of text.

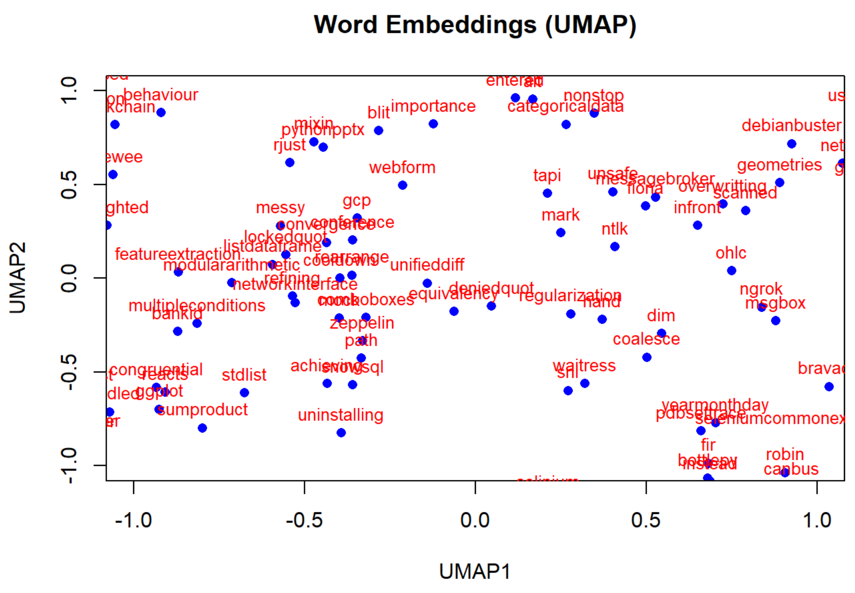
- Goal: Word2Vec is like a brain for computers that tries to understand words in a way that's similar to how humans do.
- Representation: Instead of just knowing what a word is, Word2Vec wants to represent each word as a special kind of number. These numbers capture the essence of what each word means.
- How it Learns: Imagine teaching a computer to understand words by reading a lot of text. It looks at the words that often appear together and learns that they must be related.
- Two Jobs: Word2Vec has two main jobs or games to play: Guess the Word: Given the words around it, try to guess what the missing word is. This helps the computer learn the meaning of words based on context.
- Guess the Context: Given a word, try to guess what words are likely to appear around it. This helps the computer understand how words relate to each other.
- Vectors: Once trained, each word gets its special set of numbers (vector). Words with similar meanings have vectors that point in similar directions.

Keras Word2Vec Results

Question -> How to run python in conda environment ?

```
##       run elevenlabs     pitch   layered      book      days  logstash
##  1.0000000  0.5737176 0.5550957 0.5288951 0.4849439 0.4794046 0.4791245
```

Word Embedding



Paragraph2vec Approach

- Contextual Understanding: Paragraph2Vec excels at capturing contextual information in variable-length Stack Overflow questions, providing a holistic view of the entire text.
- Document-Level Semantics: Tailored for understanding document-level semantics, Paragraph2Vec is well-suited for tasks that require comprehending the overall context and purpose of Stack Overflow questions.
- Efficient Retrieval and Similarity: Ideal for tasks like document retrieval and similarity, Paragraph2Vec embeddings facilitate efficient operations by reducing dimensionality while retaining semantic information.
- Handling Variable-Length Texts: As Stack Overflow questions vary in length, Paragraph2Vec's ability to handle variable-length texts ensures robust performance in capturing meaningful relationships within questions.

Paragraph2Vec Result

Question -> How to run python in conda environment ?

Similarity Matrix

```
##          term1        term2 similarity rank
## 1        conda     anaconda  0.9151204    1
## 2        conda environement  0.8750440    4
## 3        conda    miniconda  0.8700238    5
## 4        conda     pippython  0.8859061    3
## 5        conda        pyopt  0.8862889    2
## 6  environment          env  0.9720023    1
## 7  environment   enviroment  0.9143285    3
## 8  environment      utility  0.8447001    5
## 9  environment         venv  0.8642362    4
## 10 environment      virtual  0.9175690    2
## 11      python    overwrite  0.8554232    3
## 12      python      passing  0.8754611    1
## 13      python     printing  0.8479622    5
## 14      python      pythonx  0.8709121    2
## 15      python      writing  0.8501320    4
## 16         run      execute  0.9763917    1
## 17         run    executing  0.9418076    2
## 18         run       launch  0.9300061    5
## 19         run      running  0.9352252    4
## 20         run         runs  0.9377644    3
```

Related Tags

```
##  [1] "execute"      "env"          "executing"    "runs"         "running"
##  [6] "launch"       "virtual"      "anaconda"     "enviroment"   "pyopt"
## [11] "pippython"    "passing"      "environement" "pythonx"      "miniconda"
## [16] "venv"         "overwrite"    "writing"
```

Top 5 Tags

```
## [1] "execute"   "env"       "launch"    "virtual"   "anaconda"  "miniconda"
## [7] "overwrite"
```