Procedure for analysing FAQs in communication channels (e.g., Slack)

Dereje W. Mekonnen, webeet.io

This procedure outlines the steps required to examine Frequently Asked Questions (FAQs) within a company's communication channels, such as Slack. The methodology is based on an analysis I recently conducted for my company. For coding examples, I will use a hypothetical DataFrame name (df) with a column named “message_text”, which contains messages exchanged between employees or between an employee and a client. These messages may include questions, notifications, or other types of communication. Each message is stored as a separate row, with a unique timestamp and metadata.

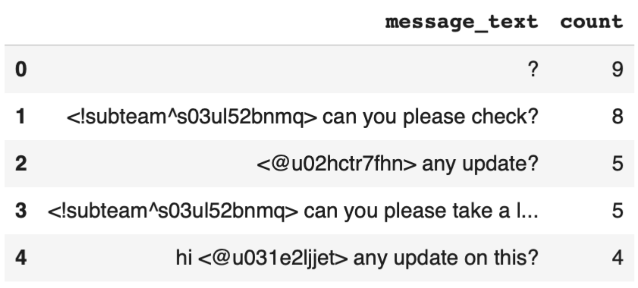
Step 1: Filter messages containing questions

Select rows that contain a question mark (?) to ensure that only messages classified as questions are extracted. In this project, more than 16000 rows of data were extracted as questions from a total of 67000 rows. This step helps differentiate questions from other types of messages, such as notifications or general statements.

```
questions = df[df["masked_text"].str.contains(r"\?", na=False)]
```

Since "?" is a special character in regex, it is not automatically recognised as a literal question mark. To ensure correct matching, it must be escaped using "\?". Using r"\?" in the regex pattern ensures that only messages containing a question mark (?) are selected. Additionally, if there are null values in the column, they will be overridden to prevent errors. The resulting questions DataFrame was used for subsequent analysis.

Example output:



```
                                      message_text  count
0                                                ?      9
1       <!subteam^s03ul52bnmq> can you please check?      8
2                       <@u02hctr7fhn> any update?      5
3  <!subteam^s03ul52bnmq> can you please take a l...      5
4               hi <@u031e2ljjet> any update on this?      4
```

Step 2: Pre-processing the “message_text” column

The “message_text” column in the filtered questions DataFrame needs to be cleaned by removing stopwords (common words like "the", "is", "and") to enhance text analysis. This step requires the use of the Natural Language Toolkit (NLTK) library.

2.1 Importing NLTK and loading stopwords

First, the nltk library was imported followed by the stopwords list from the nltk.corpus module. Lists of English stopwords are loaded from stopwords.words() and converted into set for faster lookups. Example of English stop words include (‘am', 'doing', 'is','the', 'has', 'an', 'on', 'some', 'between', ‘himself’).

```
import nltk
from nltk.corpus import stopwords
nltk.download("stopwords")
stop_words =  set(stopwords.words(“english"))
```

2.2 Cleaning the message_text column by removing stopwords

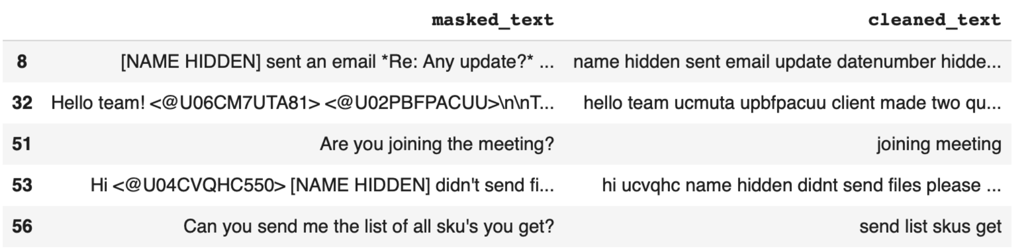
The function defined for text preprocessing follows these steps:

a. Convert text to lowercase: This ensures uniformity by converting all characters to lowercase.
b. Remove numbers and punctuation: By using regular expressions (regex), everything except letters (a-z) is replaced with an empty string. This step eliminates numbers and punctuations from the “message_text” column.
c. Tokenization : The .split() method splits the text into a list of words.
d. Stopword removal: A list comprehension filters out stopwords obtained in Section 2.1.
e. Reconstructing the text: The cleaned words are joined back into a single string.

The cleaned text is then added to the DataFrame as a new column named “cleaned_text”.

```
def clean_text(text):
   text = text.lower()
   text = re.sub(r"[^a-z\s]", "", text)  # Remove punctuation
   words = text.split()
   words = [word for word in words if word not in stop_words]
   return " “.join(words)
questions["cleaned_text"] = questions[“masked_text"].apply(clean_text)
```

Example output:



	masked_text	cleaned_text
8	[NAME HIDDEN] sent an email *Re: Any update?* ...	name hidden sent email update datenumber hidde...
32	Hello team! <@U06CM7UTA81> <@U02PBFPACUU>\n\nT...	hello team ucmuta upbfpacuu client made two qu...
51	Are you joining the meeting?	joining meeting
53	Hi <@U04CVQHC550> [NAME HIDDEN] didn't send fi...	hi ucvqhc name hidden didnt send files please ...
56	Can you send me the list of all sku's you get?	send list skus get

Step 3: Identifying the most frequent topics and their proportions

This step involves analysing the most commonly raised topics and calculating the proportion of each topic in the dataset. The process consists of several sub-steps, as outlined below.

3.1 Converting text into a token count matrix

To achieve this, CountVectorizer from the Scikit-Learn library was used. This tool processes the 'cleaned_text' column by:

a. Importing CountVectorizer: A feature extraction tool from sklearn.feature_extraction.text.

b. Learning and transforming words: The CountVectorizer scans each row of the cleaned_text column, identifies unique words, and converts them into a count matrix.

c. Filtering out extreme words: Excludes words that appear in more than 95% of the rows (too common, not useful). Ignores words that appear in fewer than 5 rows (too rare).

d. Generating a numerical matrix: Each unique word becomes a column. In this project 3986 columns were created from each word. The matrix records how many times each word appears in each row.

In essence, CountVectorizer transforms text into a structured numerical format, making it suitable for topic modelling and further analysis.

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_df=0.95, min_df=5)
X = vectorizer.fit_transform(questions["cleaned_text"])
```

Example output:

```
   aa  abastecimiento  ability  able  absence  absolute  ac  accept  acceptable  accepted  ...  yoyo  zendesk  zero  zeroed  zeroing  zeros  zip  zipped  zone
0   0               0        0     0        0         0   0       0           0         0  ...     0        0     0       0        0      0    0       0     0
1   0               0        0     1        0         0   0       0           0         0  ...     0        0     0       0        0      0    0       0     0
2   0               0        0     0        0         0   0       0           0         0  ...     0        0     0       0        0      0    0       0     0
3   0               0        0     0        0         0   0       0           0         0  ...     0        0     0       0        0      0    0       0     0
4   0               0        0     0        0         0   0       0           0         0  ...     0        0     0       0        0      0    0       0     0
5 rows × 3986 columns
```

3.2 Discovering hidden topics

To uncover the hidden topics within the dataset, an unsupervised machine learning algorithm, specifically Latent Dirichlet Allocation (LDA), was used. This method helps identify key themes across all questions.

```
lda_model = LatentDirichletAllocation(n_components=10, random_state=42)
lda_model.fit(X)

lda_model.transform(X)
```

Steps Involved:

a. Applying LDA: The Latent Dirichlet Allocation model is initialised with n_components=10, meaning it aims to identify up to 10 distinct topics.

b. Fitting the model: The LDA algorithm is applied to the count matrix generated in Step 3.1, learning the distribution of words within topics.

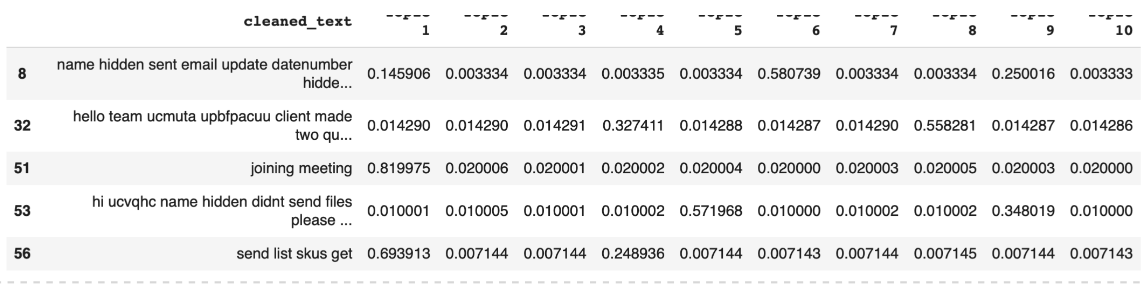
c. Extracting top words: The model determines the most relevant words for each topic, helping interpret the nature of each theme

Example output:

```
Topic 1:
['want', 'share', 'hiddenlf', 'team', 'time', 'need', 'client', 'hi', 'unumber', 'meeting']


Topic 2:
['issue', 'st', 'today', 'update', 'vis', 'thanks', 'ucbhnfl', 'hi', 'hiddenej', 'unsnumber']


Topic 3:
['inventory', 'product', 'location', 'hiddenthreadtsnumber', 'sku', 'id', 'hiddennumber', 'url', 'number', 'hidden']


Topic 4:
['skus', 'ucbhnfl', 'locations', 'dont', 'inventories', 'location', 'ar', 'inventory', 'need', 'gt']
```

d. Assigning topics to each question. Each question (row in the cleaned_text column) is assigned a probability distribution over the discovered topics. The highest probability value determines the primary topic for that question.

Example Output:



	cleaned_text	1	2	3	4	5	6	7	8	9	10
8	name hidden sent email update datenumber hidde...	0.145906	0.003334	0.003334	0.003335	0.003334	0.580739	0.003334	0.003334	0.250016	0.003333
32	hello team ucmuta upbfpacuu client made two qu...	0.014290	0.014290	0.014291	0.327411	0.014288	0.014287	0.014290	0.558281	0.014287	0.014286
51	joining meeting	0.819975	0.020006	0.020001	0.020002	0.020004	0.020000	0.020003	0.020005	0.020003	0.020000
53	hi ucvqhc name hidden didnt send files please ...	0.010001	0.010005	0.010001	0.010002	0.571968	0.010000	0.010002	0.010002	0.348019	0.010000
56	send list skus get	0.693913	0.007144	0.007144	0.248936	0.007144	0.007143	0.007144	0.007145	0.007144	0.007143

e. Identifying the most frequent topics:

- The number of questions assigned to each topic was counted. To achieve this, each row was first labeled with its dominant topic based on the highest probability value obtained in the previous step. Then, the total number of rows within each topic was calculated using the value_counts() function in Python.

```
questions["dominant_topic"] = np.argmax(topic_distribution, axis=1)
topic_counts = questions["dominant_topic"].value_counts()
```

- This helps identify the most frequently asked topics in the communication channel.

By following this approach, one can gain insights into what employees or clients ask about most frequently and categorise similar discussions for further analysis.

Step 4: Assigning a theme or representative word to each topic

In Section 3.2, the most frequently occurring words within each topic was identified. Based on these words, one can assign a meaningful theme to each topic.

There are two main approaches to assigning themes:

a. Manual Assignment: By reviewing the top words for each topic and assigning a relevant theme based on personal interpretation.

b. AI-Assisted Assignment: By using Large Language Models (LLMs) such as ChatGPT to analyse the top words and suggest an appropriate theme.

In this project, themes such as "Technical Issues", "Follow-ups", and "Troubleshooting" were assigned based on LLM interpretation of the topic words. This step ensures that each topic is easily understandable and aligned with business needs.