Homework #4. Exploratory Data Analysis

Author: `ANDRII VALENIA`

Intermediate result time spent on h/w (in minutes): `300`

Total time spent on h/w (in minutes): `–`

Data analysis would be performed on the dataset containing information about chatgpt conversations.

In [2]:
```
%pip install -r requirements.txt
```

```
Requirement already satisfied: pandas==2.2.3 in ./venv/lib/python3.11/site
-packages (from -r requirements.txt (line 1)) (2.2.3)
Requirement already satisfied: numpy==2.1.2 in ./venv/lib/python3.11/site-
packages (from -r requirements.txt (line 2)) (2.1.2)
Requirement already satisfied: matplotlib==3.9.2 in ./venv/lib/python3.11/
site-packages (from -r requirements.txt (line 3)) (3.9.2)
Requirement already satisfied: langdetect==1.0.9 in ./venv/lib/python3.11/
site-packages (from -r requirements.txt (line 4)) (1.0.9)
Requirement already satisfied: tqdm==4.66.6 in ./venv/lib/python3.11/site-
packages (from -r requirements.txt (line 5)) (4.66.6)
Requirement already satisfied: tabulate==0.9.0 in ./venv/lib/python3.11/si
te-packages (from -r requirements.txt (line 6)) (0.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in ./venv/lib/python
3.11/site-packages (from pandas==2.2.3->-r requirements.txt (line 1)) (2.
9.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./venv/lib/python3.11/site-
packages (from pandas==2.2.3->-r requirements.txt (line 1)) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in ./venv/lib/python3.11/sit
e-packages (from pandas==2.2.3->-r requirements.txt (line 1)) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in ./venv/lib/python3.11/s
ite-packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (1.3.
0)
Requirement already satisfied: cycler>=0.10 in ./venv/lib/python3.11/site-
packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./venv/lib/python3.11/
site-packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (4.5
4.1)
Requirement already satisfied: kiwisolver>=1.3.1 in ./venv/lib/python3.11/
site-packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (1.4.
7)
Requirement already satisfied: packaging>=20.0 in ./venv/lib/python3.11/si
te-packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (24.1)
Requirement already satisfied: pillow>=8 in ./venv/lib/python3.11/site-pac
kages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./venv/lib/python3.11/s
ite-packages (from matplotlib==3.9.2->-r requirements.txt (line 3)) (3.2.
0)
Requirement already satisfied: six in ./venv/lib/python3.11/site-packages
(from langdetect==1.0.9->-r requirements.txt (line 4)) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Loading data

In [3]:
```
import pandas as pd

dataset_path = 'data/chatlogs-v2.jsonl'

df = pd.read_json(dataset_path, lines=True)
```

In [206…
```
from tabulate import tabulate
df_copy = df.copy()
df_copy['conversation'] = df_copy['conversation'].apply(lambda x: ' '.joi
print(tabulate(df_copy.head(), headers='keys', tablefmt='psql'))
del df_copy
```

```
+----+---------------+-----------------+-------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------+-----------------+
|    | post_number   | system_message   | conversation
|   message_count |
|----+---------------+-----------------+-------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------+-----------------|
|  0 | [57]          | ['']             | What is co Contrastiv Do we need
In contras Right, but Yes, that' What is th The best a What are t Triplet
lo
|              10 |
|  1 | [53]          | ['']             | write a po John Kaerc
|               2 |
|  2 | [56]          | ['']             | What is co Contrastiv Do we need
In contras Right, but Yes, that' What is th The best a What are t Triplet
lo What kind  CLIP (Cont Can you ex Instance d Is instanc It is gene How d
o you To sample |              18 |
|  3 | [54]          | ['']             | Can we use As an AI l
|               2 |
|  4 | [51]          | ['']             | como eu fa Para ler u
|               2 |
+----+---------------+-----------------+-------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------+-----------------+
```

Let's transform the data to a more convenient format where each row corresponds to a single conversation message.

In [ ]:
```
rows = []
for _, row in df.iterrows():
    post_number = row['post_number']
    conversation = row['conversation']

    if isinstance(conversation, list):
        for message_order, message in enumerate(conversation):
            user = message['user']
            text = message['message']
            rows.append({
                'post_number': post_number,
                'user': user,
                'message': text,
                'message_order': message_order
```

```
            })

df_expanded = pd.DataFrame(rows)
```

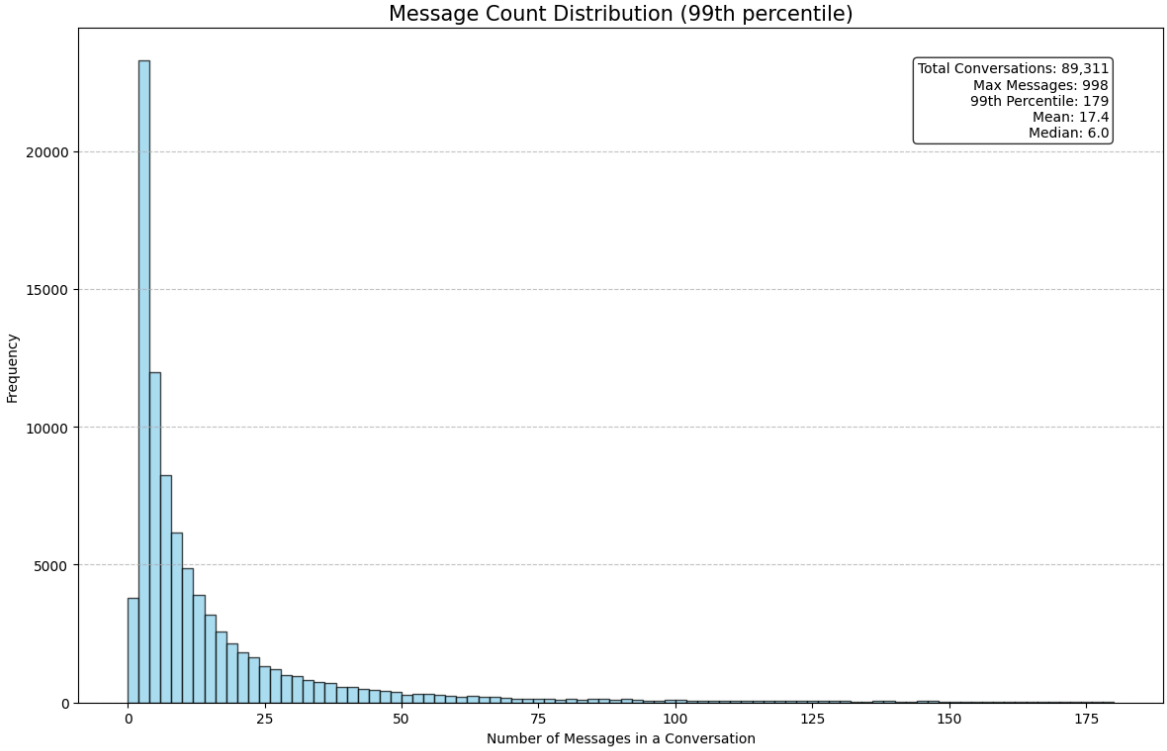
In [ ]:
```
df_expanded_truncated = df_expanded.copy()
df_expanded_truncated['message'] = df_expanded_truncated['message'].str[:
print(tabulate(df_expanded_truncated.head(), headers='keys', tablefmt='ps
print("Messages count:", len(df_expanded))
del df_expanded_truncated
```

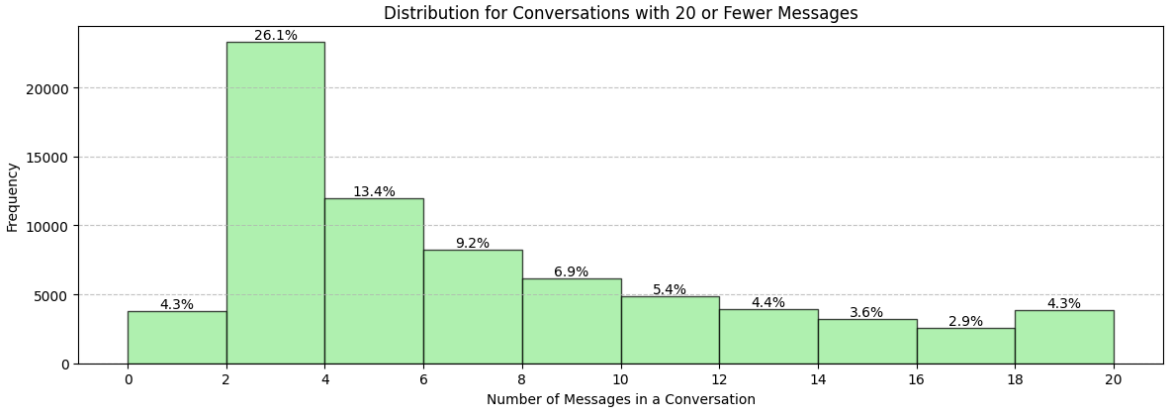
```
+----+--------------+----------+------------------------+-------------
---+
|    | post_number   | user      | message                |   message_ord
er |
|----+--------------+----------+------------------------+-------------
---|
|  0 | [57]          | Anonymous | What is contrastive ... |
0 |
|  1 | [57]          | Chat GPT  | Contrastive loss is ... |
1 |
|  2 | [57]          | Anonymous | Do we need labels fo... |
2 |
|  3 | [57]          | Chat GPT  | In contrastive learn... |
3 |
|  4 | [57]          | Anonymous | Right, but we still ... |
4 |
+----+--------------+----------+------------------------+-------------
---+
Messages count: 1549625
```

General conversation analysis

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

df['message_count'] = df['conversation'].apply(lambda x: len(x) if isinst

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12), height_ratios=[2,

max_percentile = np.percentile(df['message_count'], 99)
bins = np.arange(0, max_percentile + 2, 2)

ax1.hist(df['message_count'][df['message_count'] <= max_percentile],
         bins=bins,
         color='skyblue',
         edgecolor='black',
         alpha=0.7)

ax1.set_title('Message Count Distribution (99th percentile)', fontsize=15
ax1.set_xlabel('Number of Messages in a Conversation', fontsize=10)
ax1.set_ylabel('Frequency', fontsize=10)
ax1.grid(axis='y', linestyle='--', alpha=0.7)

stats_text = f'Total Conversations: {len(df):,}\n'
stats_text += f'Max Messages: {df["message_count"].max():,}\n'
stats_text += f'99th Percentile: {max_percentile:.0f}\n'
stats_text += f'Mean: {df["message_count"].mean():.1f}\n'
stats_text += f'Median: {df["message_count"].median():.1f}'
```

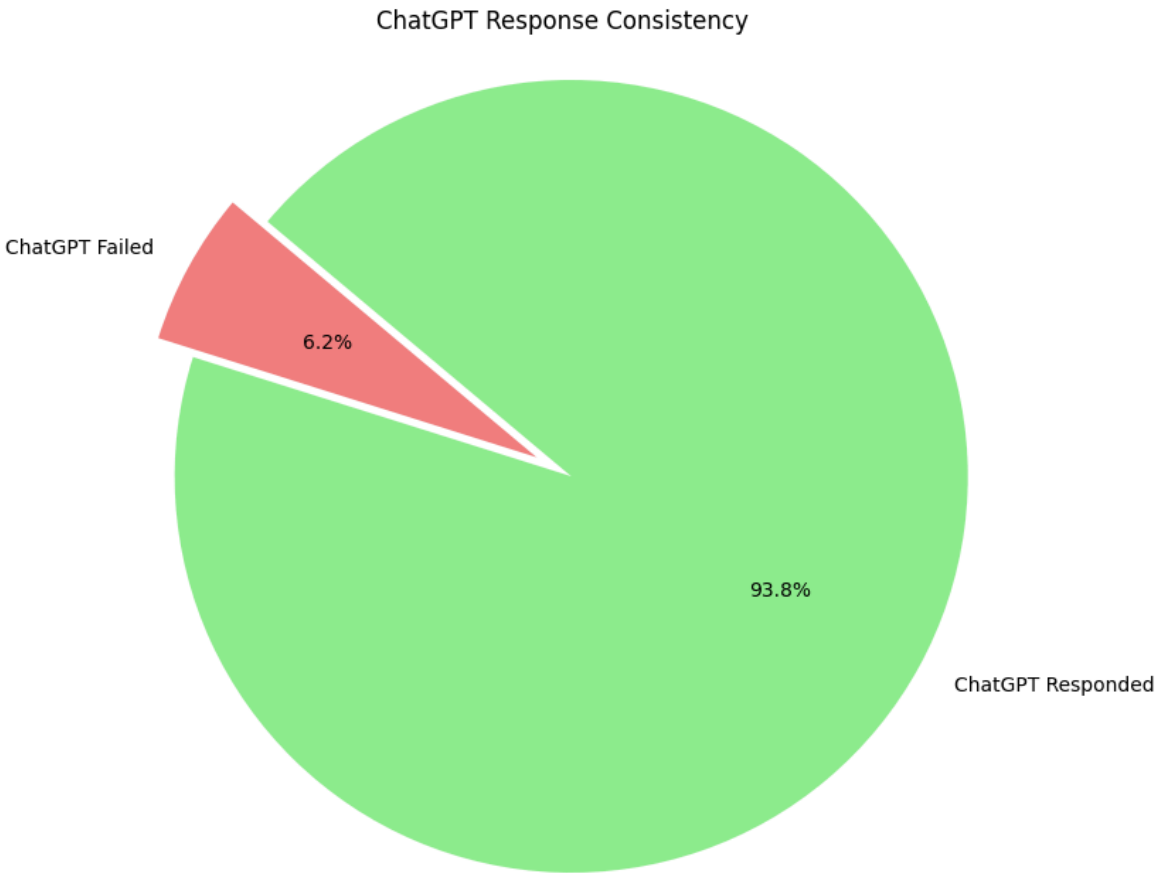
```
ax1.text(0.95, 0.95, stats_text,
         transform=ax1.transAxes,
         verticalalignment='top',
         horizontalalignment='right',
         bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

small_plot_size = 20
bins_small = np.arange(0, small_plot_size + 2, 2)
counts, bins, patches = ax2.hist(df['message_count'][df['message_count']
                                 bins=bins_small,
                                 color='lightgreen',
                                 edgecolor='black',
                                 alpha=0.7)

total = len(df)
for i, count in enumerate(counts):
    percentage = (count / total) * 100
    ax2.text(bins[i] + 1, count, f'{percentage:.1f}%',
             ha='center', va='bottom')

ax2.set_title('Distribution for Conversations with 20 or Fewer Messages',
ax2.set_xlabel('Number of Messages in a Conversation', fontsize=10)
ax2.set_ylabel('Frequency', fontsize=10)
ax2.grid(axis='y', linestyle='--', alpha=0.7)
ax2.set_xticks(bins_small)

plt.tight_layout()
plt.show()
```





In [203…

```
last_message_user = df_expanded.groupby('post_number').last()['user']
chatgpt_responded = last_message_user == 'Chat GPT'

chatgpt_responded_count = chatgpt_responded.sum()
chatgpt_failed_count = len(chatgpt_responded) - chatgpt_responded_count

labels = ['ChatGPT Failed', 'ChatGPT Responded']
sizes = [chatgpt_failed_count, chatgpt_responded_count]
colors = ['lightcoral', 'lightgreen']

plt.figure(figsize=(10, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangl
plt.title('ChatGPT Response Consistency')
plt.figtext(0.5, 0.01, 'Conversations where the last message is from Chat
plt.axis('equal')
plt.show()
```



Conversations where the last message is from ChatGPT indicate a successful response.

Language analysis

Detect the language of the messages.

In [62]:
```
from concurrent.futures import ThreadPoolExecutor, as_completed
from langdetect import detect, LangDetectException
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from time import time

def safe_language_detect(text) -> str:
    try:
        return detect(str(text))
    except LangDetectException:
        return 'unknown'

def process_batch(texts):
    return [safe_language_detect(text) for text in texts]

def create_batches(df, batch_size=100) -> list:
    return [df['message'].iloc[i:i + batch_size].tolist()
            for i in range(0, len(df), batch_size)]

def detect_languages(df, batch_size=100, max_workers=12) -> list:
    batches = create_batches(df, batch_size)
    results = []
```

```
    total_batches = len(batches)
    processed_batches = 0
    start_time = time()

    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        futures = [executor.submit(process_batch, batch) for batch in bat

        for future in as_completed(futures):
            batch_results = future.result()
            results.extend(batch_results)
            processed_batches += 1

            progress = processed_batches / total_batches
            elapsed_time = time() - start_time
            est_total_time = elapsed_time / progress if progress > 0 else
            est_remaining = est_total_time - elapsed_time
            print(f"\rProgress: {progress:.1%} | "
                  f"Processed: {processed_batches}/{total_batches} batc
                  f"Est. remaining: {est_remaining:.1f}s", end="")

    print("\nProcessing completed!")
    return results

from pathlib import Path
csv_path = 'data/expanded_chatlogs.csv'

if Path(csv_path).is_file():
    df_expanded = pd.read_csv(csv_path)
    print("Using existing CSV file.")
else:
    print("Detecting languages for each message...")
    df_expanded['language'] = detect_languages(df_expanded)
    df_expanded.to_csv(csv_path, index=False)
```

```
Using existing CSV file.
```

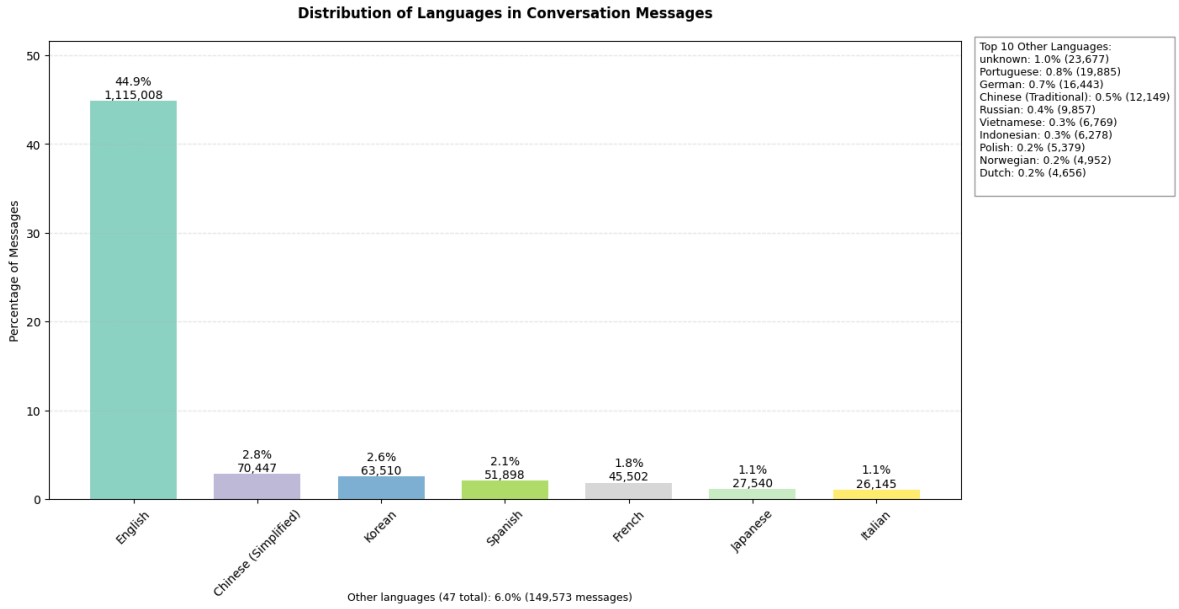
```
/var/folders/y5/0t8gc4cj315cwfcx9vxy3kgc0000gn/T/ipykernel_94916/256115149
0.py:51: DtypeWarning: Columns (1,2,3) have mixed types. Specify dtype opt
ion on import or set low_memory=False.
  df_expanded = pd.read_csv(csv_path)
```

Took 54 minutes to detect language of each message.

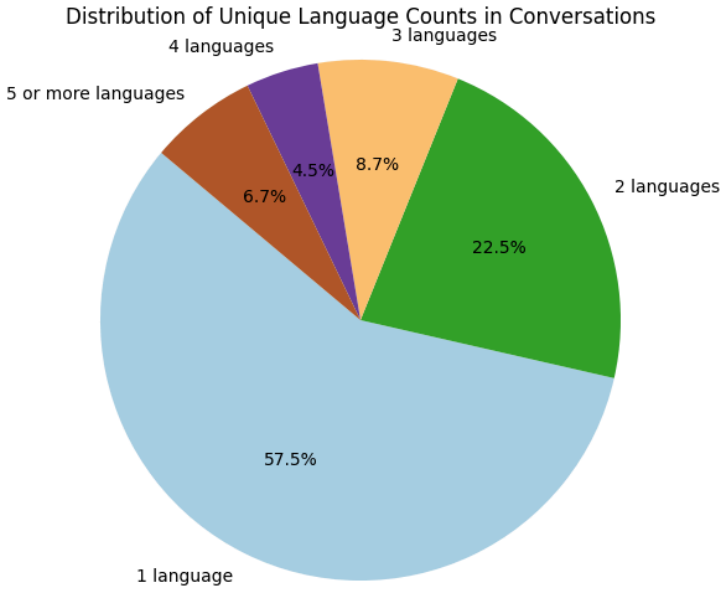
In [ ]:

```
import pycountry

def get_country_name(code):
    if code == 'zh-cn':
        return 'Chinese (Simplified)'
    elif code == 'zh-tw':
        return 'Chinese (Traditional)'
    country_info = pycountry.languages.get(alpha_2=code)
    return country_info.name if country_info else code

def plot_language_distribution(df, threshold_percent=2):
    language_counts = df['language'].value_counts()
    total_messages = len(df)

    percentages = (language_counts / total_messages) * 100
    percentages_sorted = percentages.sort_values(ascending=False)
    main_languages = percentages_sorted[percentages_sorted >= threshold_p
    other_languages = percentages_sorted[percentages_sorted < threshold_p
```

```
    plt.figure(figsize=(14, 8))

    main_ax = plt.gca()
    main_ax.bar(range(len(main_languages)),
                main_languages,
                color=plt.cm.Set3(np.linspace(0, 1, len(main_languages)))
                width=0.7)

    for i, v in enumerate(main_languages):
        count = int(v * total_messages / 100)
        main_ax.text(i, v, f'{v:.1f}%\n{count:,}',
                     ha='center', va='bottom', fontsize=10)

    main_ax.set_xticks(range(len(main_languages)))
    main_ax.set_xticklabels([get_country_name(lang) for lang in main_lang
    main_ax.set_ylabel('Percentage of Messages', fontsize=10)
    main_ax.set_ylim(0, max(main_languages) * 1.15)

    main_ax.set_title('Distribution of Languages in Messages',
                      pad=20, size=12, weight='bold')

    main_ax.grid(axis='y', linestyle='--', alpha=0.3)

    other_pct = other_languages.sum()
    other_count = int(other_pct * total_messages / 100)

    top_10_text = "Top 10 Other Languages:\n"
    for lang, pct in other_languages[:10].items():
        count = int(pct * total_messages / 100)
        top_10_text += f"{get_country_name(lang)}: {pct:.1f}% ({count:,})

    main_ax.text(1.02, 1.0, top_10_text,
                 transform=main_ax.transAxes,
                 bbox=dict(facecolor='white', edgecolor='gray', alpha=0.8
                 fontsize=9,
                 verticalalignment='top')

    summary_text = f'Other languages ({len(other_languages)} total): {oth
    plt.figtext(0.5, 0.05, summary_text, ha='center', fontsize=9)
    plt.subplots_adjust(bottom=0.2)
    plt.show()

plot_language_distribution(df_expanded, threshold_percent=1)
```



In [ ]:
```
language_counts_per_conversation = df_expanded.groupby('post_number')['la
language_counts_per_conversation.columns = ['post_number', 'unique_langua
language_counts_per_conversation['unique_language_count'] = language_coun
language_count_distribution = language_counts_per_conversation['unique_la
labels = [f'{count} language{"s" if count != "1" else ""}' for count in l

plt.figure(figsize=(12, 6))
plt.pie(language_count_distribution,
    labels=labels,
    autopct='%1.1f%%',
    colors=plt.cm.Paired(np.linspace(0, 1, len(language_count_distributio
    startangle=140)
plt.title('Distribution of Unique Language Counts in Conversations')
plt.axis('equal')
plt.show()
```



In [116…
```
df_temp = df_expanded.copy()
df_temp['next_user'] = df_temp.groupby('post_number')['user'].shift(-1)
df_temp['next_language'] = df_temp.groupby('post_number')['language'].shi

pairs_df = df_temp[
    (df_temp['user'] != 'Chat GPT') &
```

```
    (df_temp['next_user'] == 'Chat GPT')
].copy()

pairs_df = pairs_df[['post_number', 'language', 'next_language']].rename(
    columns={'language': 'user_language', 'next_language': 'chatgpt_langu
)
pairs_df['pair_order'] = pairs_df.groupby('post_number').cumcount()
pairs_df
```

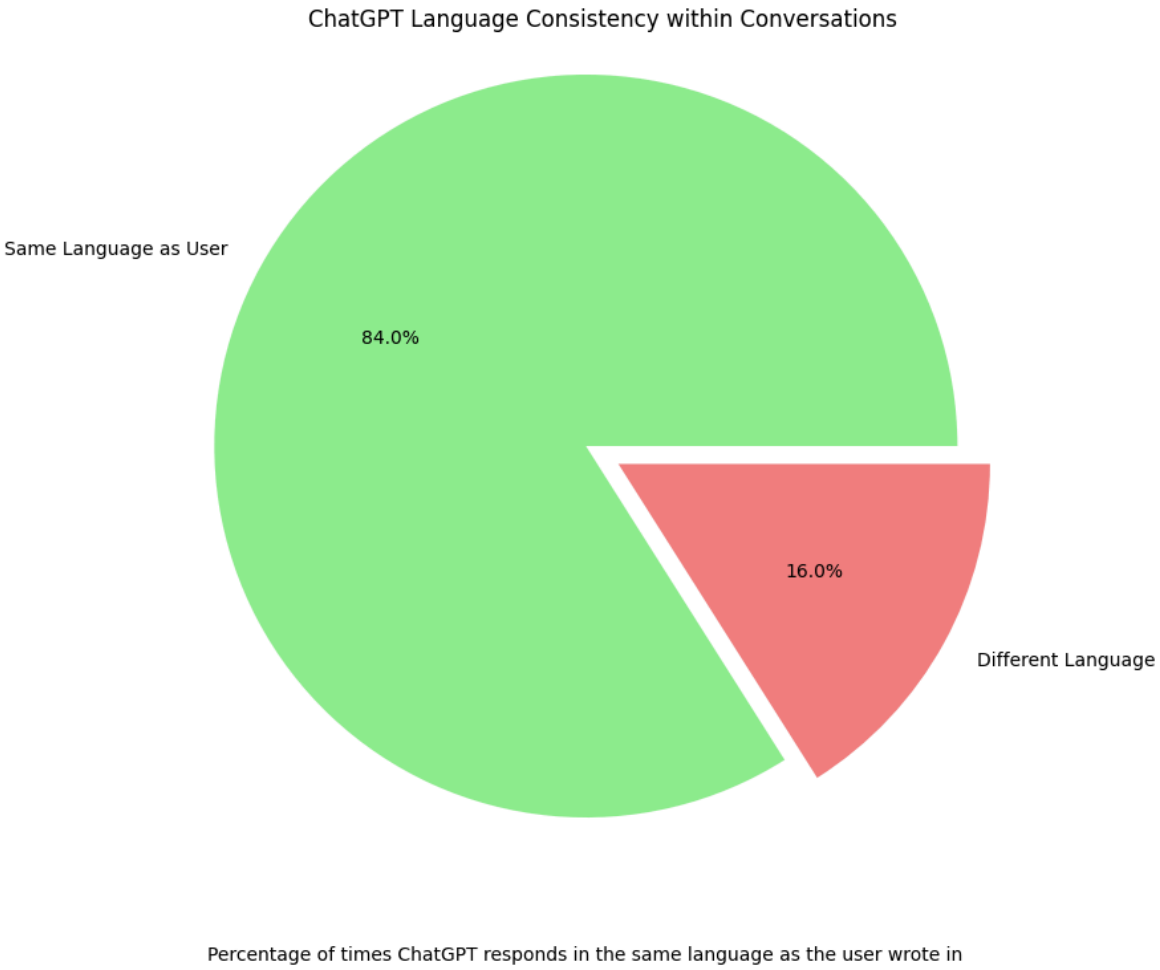
Out[116…

	post_number	user_language	chatgpt_language	pair_order
0	[57]	unknown	unknown	0
2	[57]	unknown	unknown	1
4	[57]	unknown	unknown	2
6	[57]	unknown	unknown	3
8	[57]	unknown	unknown	4
…	…	…	…	…
2484151	[22628]	en	en	73
2484153	[22628]	en	en	74
2484155	[22628]	en	en	75
2484157	[22628]	en	en	76
2484160	[22628]	en	en	77

758234 rows × 4 columns

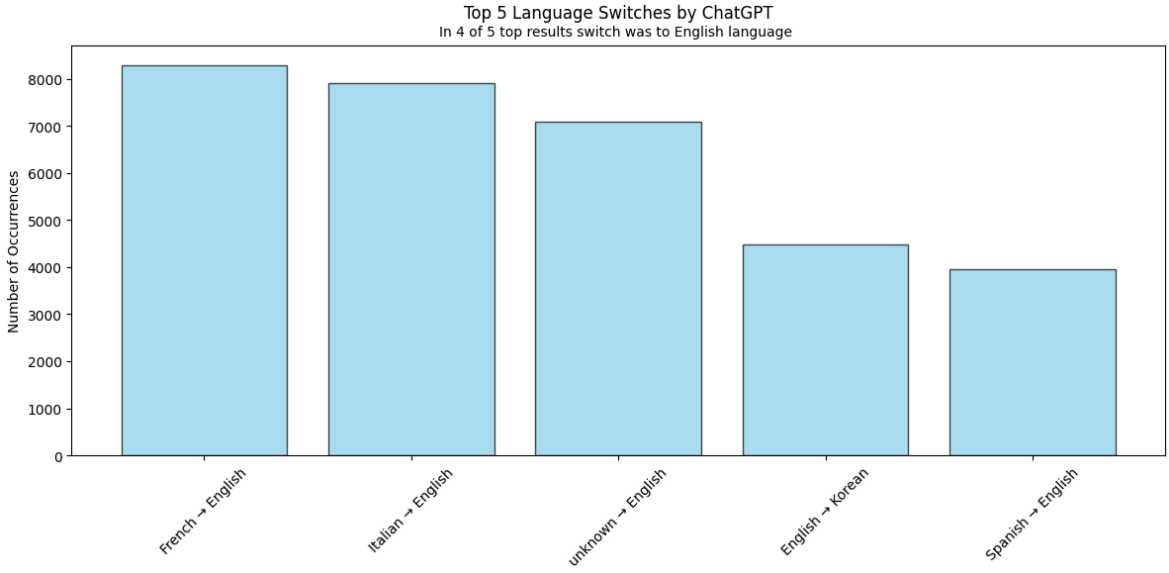
In [192…

```
total_pairs = len(pairs_df)
matching_languages = (pairs_df['user_language'] == pairs_df['chatgpt_lang
percentage_matching = (matching_languages / total_pairs) * 100

plt.figure(figsize=(10, 8))
plt.pie([percentage_matching, 100-percentage_matching],
        labels=['Same Language as User', 'Different Language'],
        colors=['lightgreen', 'lightcoral'],
        autopct='%1.1f%%',
        explode=(0.1, 0))
plt.title('ChatGPT Language Consistency within Conversations')
plt.figtext(0.5, 0.01, 'Percentage of times ChatGPT responds in the same
plt.axis('equal')

plt.show()
```



In [186…

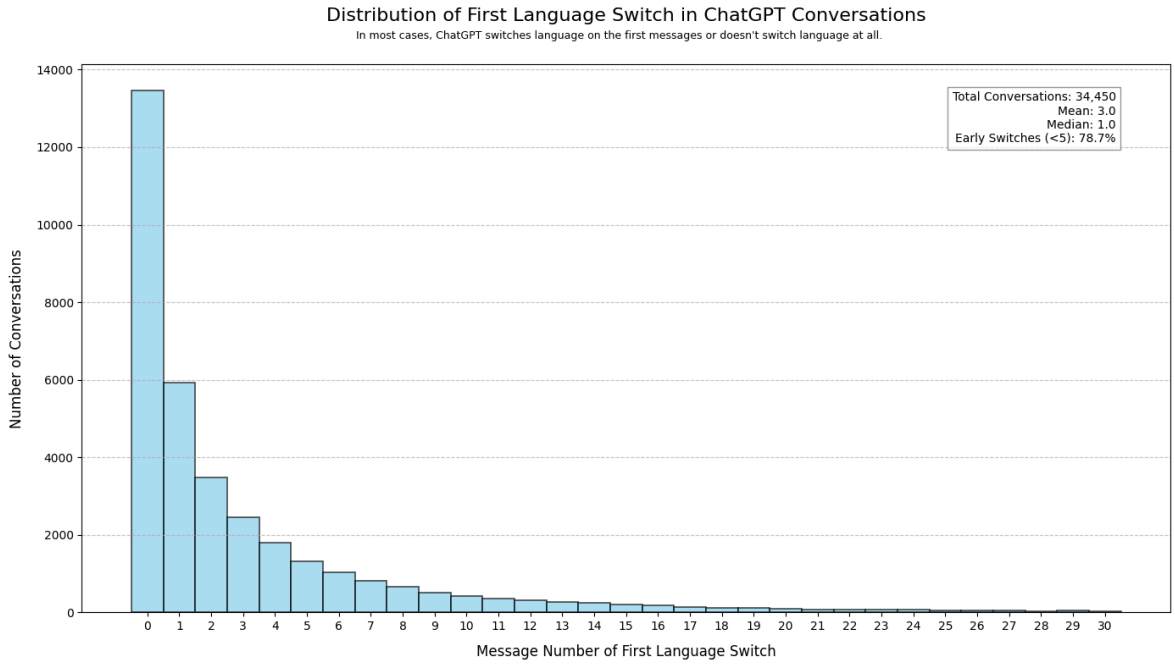
```
different_lang_responses = pairs_df[pairs_df['user_language'] != pairs_df

language_switches = different_lang_responses.groupby(['user_language', 'c
language_switches.columns = ['User Language', 'ChatGPT Response', 'Count'
language_switches = language_switches.sort_values('Count', ascending=Fals

plt.figure(figsize=(12, 6))
top_5_switches = language_switches.head(5)
plt.bar(range(len(top_5_switches)), top_5_switches['Count'], color='skybl
plt.xticks(range(len(top_5_switches)),
           [f'{get_country_name(row["User Language"])} → {get_country_name
            for _, row in top_5_switches.iterrows()],
           rotation=45)
plt.title('Top 5 Language Switches by ChatGPT', pad=20)
plt.figtext(0.525, 0.94, 'In 4 of 5 top results switch was to English lan
plt.ylabel('Number of Occurrences')
plt.tight_layout()
plt.show()
```



In [187…

```
first_mismatches = different_lang_responses.groupby('post_number')[
    'pair_order'].min()
first_mismatches_series = first_mismatches.value_counts().sort_index()
upper_bound = np.percentile(first_mismatches, 99)
filtered_mismatches = first_mismatches[first_mismatches <= upper_bound]

mean_switch = filtered_mismatches.mean()
median_switch = filtered_mismatches.median()
total_samples = len(filtered_mismatches)

fig, ax = plt.subplots(figsize=(14, 8))

max_message = int(np.ceil(filtered_mismatches.max()))
bins = np.arange(0, max_message + 2) - 0.5

n, bins, patches = plt.hist(filtered_mismatches,
                            bins=bins,
                            color='skyblue',
                            edgecolor='black',
                            alpha=0.7,
                            linewidth=1.2)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.title('Distribution of First Language Switch in ChatGPT Conversations
          fontsize=16, pad=20)
plt.xlabel('Message Number of First Language Switch', fontsize=12, labelp
plt.ylabel('Number of Conversations', fontsize=12, labelpad=10)

stats_text = (f'Total Conversations: {total_samples:,}\n'
              f'Mean: {mean_switch:.1f}\n'
              f'Median: {median_switch:.1f}\n'
              f'Early Switches (<5): {len(filtered_mismatches[filtered_mi
plt.text(0.95, 0.95, stats_text,
         transform=ax.transAxes,
         bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray'),
         verticalalignment='top',
         horizontalalignment='right',
         fontsize=10)

plt.figtext(0.525, 0.935,
            'In most cases, ChatGPT switches language on the first messag
```

```
            ha='center', fontsize=9)

plt.xticks(np.arange(0, max_message + 1, 1))

plt.tight_layout()
plt.show()
```



TDA: Sentiment Analysis

TDA: Topic Categorization

TDA: User Satisfaction Rate

TDA: Identify prompt techniques and how user satisfaction rate is affected by them.