03_eda

October 12, 2025

1 Step 3 — Cleaned EDA + Preparation

Goal: Clean the raw data into an analysis-ready dataset and verify the results.

What this notebook does

- Loads `pipeline datasets`.
- Cleans text: lowercasing, URL/user/emoji removal, punctuation/whitespace fixes, optional stopword removal.
- Normalizes fields (e.g., datetime, subreddit/category labels) and removes remaining duplicates/empties.
- Validates distributions again (lengths, tokens, top terms) to confirm the cleaning worked.

Inputs: `pipeline datasets`.
Outputs: Saves the cleaned corpus to `cleaned pipeline datasets`.

[ ]:
```
#
try:
    IS_PIPELINE_RUN
except NameError:
    IS_PIPELINE_RUN = False

try:
    IS_PIPELINE_TEST
except NameError:
    IS_PIPELINE_TEST = False
```

[ ]:
```
import pandas as pd
from datetime import datetime
import re
from pathlib import Path
from text_processing import (
    clean_dataframe_column,
    get_word_counts,
    get_bigram_counts,
    get_post_statistics,
)
```

```
[nltk_data] Downloading package stopwords to /home/junc/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

1.1 Process datasets

[2]:
```
dataset_folder = Path("../datasets")
datasets = {}
```

[ ]:
```
# set filenames for import
if IS_PIPELINE_RUN:
    filename1 = "Palo_Alto_pipeline_reddit.pkl"
    filename2 = "Oklahoma_City_pipeline_reddit.pkl"
else:
    filename1 = "Palo_Alto_20251007_235943_reddit.pkl"
    filename2 = "Oklahoma_City_20251008_000300_reddit.pkl"
```

[ ]:
```
# Dataset 1
df = pd.read_pickle(dataset_folder / filename1)
df = clean_dataframe_column(df, column="comments_flat")
df = get_post_statistics(df)
datasets["dataset1"] = df
print(f"dataset1: {len(df)} posts")
```

```
dataset1: 79 posts
```

[ ]:
```
# Dataset 2
df = pd.read_pickle(dataset_folder / filename2)
df = clean_dataframe_column(df, column="comments_flat")
df = get_post_statistics(df)
datasets["dataset2"] = df
print(f"dataset2: {len(df)} posts")
```

```
dataset2: 135 posts
```

1.2 Compare statistics across datasets

[5]:
```
for name, df in datasets.items():
    total_comments = df["cleaned_comments"].apply(len).sum()
    print(f"{name}:")
    print(f"  Total posts: {len(df)}")
    print(f"  Total comments: {total_comments}")
    print(f"  Average comment length: {df['avg_comment_length'].mean():.1f}␣
 ↪words")
    print(f"  Total tokens: {df['total_tokens'].sum()}")
    print()
```

```
dataset1:
  Total posts: 79
  Total comments: 3308
  Average comment length: 28.2 words
  Total tokens: 91043
```

```
dataset2:
  Total posts: 135
  Total comments: 10010
  Average comment length: 22.0 words
  Total tokens: 228638
```

1.3 Top 5 words by dataset

[6]:
```python
for name, df in datasets.items():
    all_comments = []
    for comments_list in df["cleaned_comments"]:
        all_comments.extend(comments_list)

    words = get_word_counts(all_comments, top_n=5)
    print(f"{name}:")
    print(words)
    print()
```

```
dataset1:
     word  count
0  people    775
1    like    688
2  school    611
3     get    507
4   would    487

dataset2:
     word  count
0    like   2063
1  people   1680
2     one   1328
3   would   1318
4     get   1161
```

1.3.1 Save Datasets with Cleaned Column

[ ]:
```python
def save_dataset(df, dataset_name, dataset_folder):
    """
    Save a single DataFrame using the district extracted from df['query'] as␣
 ↪filename prefix.
    """
    # extract district from first query (text inside quotes)
    if "query" in df.columns and not df["query"].empty:
        m = re.search(r'"([^"]+)"', str(df["query"].iloc[0]))
        district = m.group(1) if m else "unknown_district"
    else:
```

```
        district = "unknown_district"

    # sanitize
    district = re.sub(r"[^A-Za-z0-9_-]+", "_", district)

    # ensure folder
    folder = Path(dataset_folder or ".")
    folder.mkdir(parents=True, exist_ok=True)

    # filename
    if IS_PIPELINE_RUN:
        ts = "pipeline"
    else:
        ts = datetime.now().strftime("%Y%m%d_%H%M%S")

    path = folder / f"{district}_cleaned_{ts}_reddit.pkl"

    df.to_pickle(path)
    print(f"  Saved {dataset_name} → {path.name}")
    return path
```

[8]:
```
for name, dataset in datasets.items():
    save_dataset(dataset, name, dataset_folder)
```

```
Saved dataset1 → Palo_Alto_cleaned_20251008_005822_reddit.pkl
Saved dataset2 → Oklahoma_City_cleaned_20251008_005822_reddit.pkl
```