Introduction

This project involves creating a search engine using PyTerrier and Flask, hosted locally with ngrok for accessibility. The goal is to implement a robust text retrieval system that supports query processing and displays results through a web interface.

Data Collection

Dataset Description

The dataset comprises tweets extracted from Twitter, containing varied text data that is typical of social media.

Preprocessing

The text data was preprocessed through several steps:

Tokenization: Splitting text into individual terms.

-Cleaning: Removing URLs, special characters, and handling casing.

- Stemming: Reducing words to their root form using the Porter Stemmer

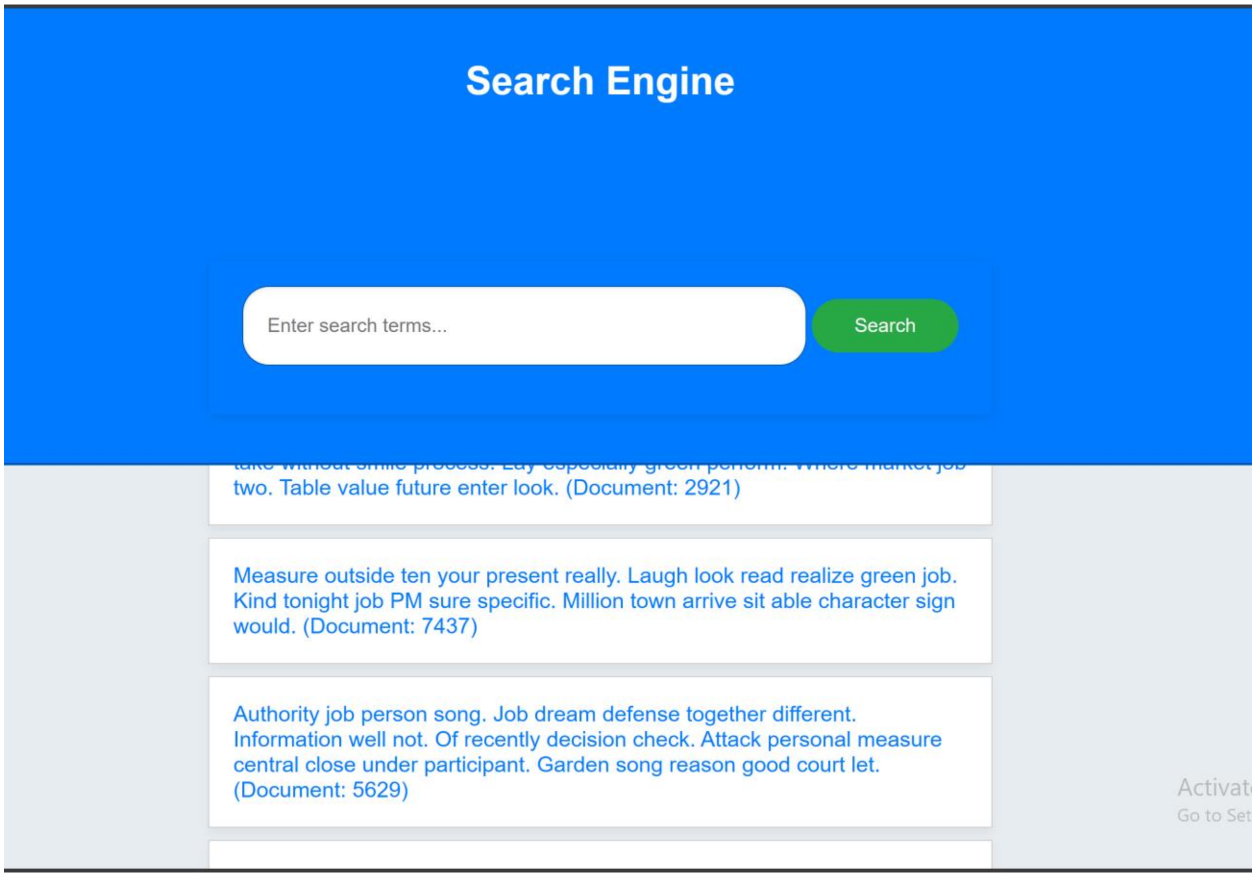
Indexing

Using PyTerrier, documents were indexed to facilitate efficient retrieval. The indexing process involved creating an inverted index where each unique term is linked to the documents it appears in, along with the term frequency.

Query Processing

Query processing mirrors the preprocessing steps applied to the document corpus to ensure consistency between the query terms and indexed terms.

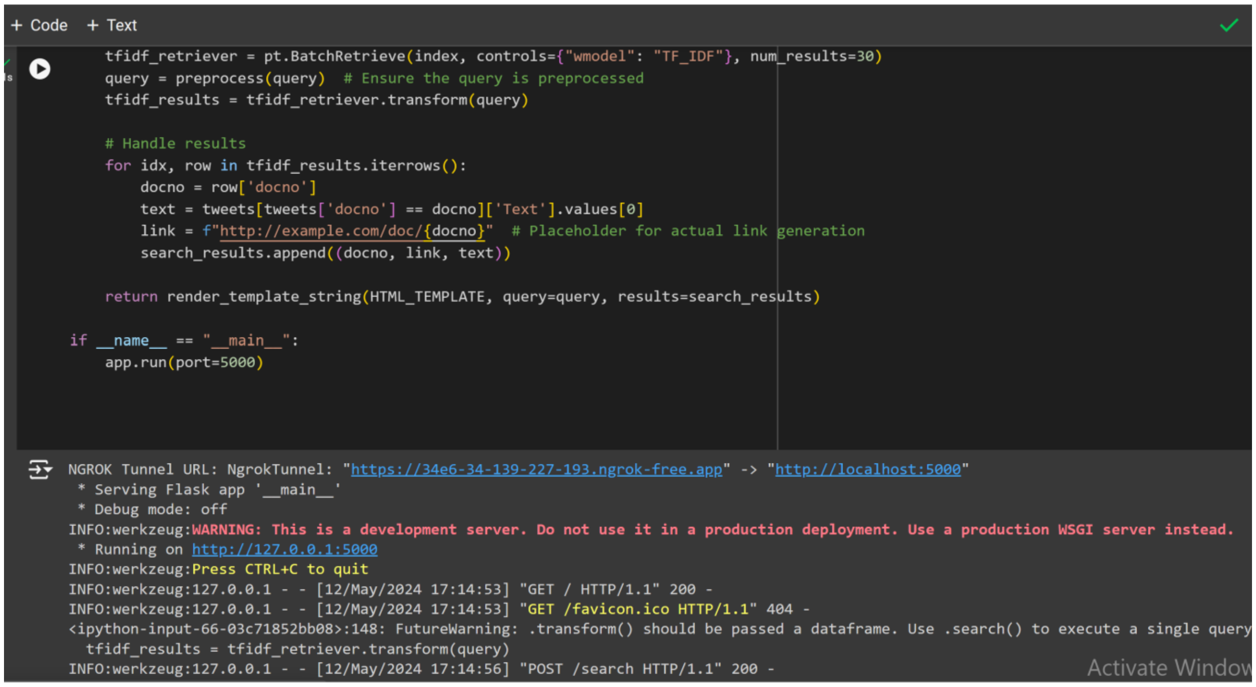
This standardization is crucial for effective matching during the search process.



two. Table value future enter look. (Document: 2921)

Measure outside ten your present really. Laugh look read realize green job. Kind tonight job PM sure specific. Million town arrive sit able character sign would. (Document: 7437)

Authority job person song. Job dream defense together different. Information well not. Of recently decision check. Attack personal measure central close under participant. Garden song reason good court let. (Document: 5629)



```
    tfidf_retriever = pt.BatchRetrieve(index, controls={"wmodel": "TF_IDF"}, num_results=30)
    query = preprocess(query)  # Ensure the query is preprocessed
    tfidf_results = tfidf_retriever.transform(query)

    # Handle results
    for idx, row in tfidf_results.iterrows():
        docno = row['docno']
        text = tweets[tweets['docno'] == docno]['Text'].values[0]
        link = f"http://example.com/doc/{docno}"  # Placeholder for actual link generation
        search_results.append((docno, link, text))

    return render_template_string(HTML_TEMPLATE, query=query, results=search_results)

if __name__ == "__main__":
    app.run(port=5000)
```

```
NGROK Tunnel URL: NgrokTunnel: "https://34e6-34-139-227-193.ngrok-free.app" -> "http://localhost:5000"
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [12/May/2024 17:14:53] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [12/May/2024 17:14:53] "GET /favicon.ico HTTP/1.1" 404 -
<ipython-input-66-03c71852bb08>:148: FutureWarning: .transform() should be passed a dataframe. Use .search() to execute a single query
  tfidf_results = tfidf_retriever.transform(query)
INFO:werkzeug:127.0.0.1 - - [12/May/2024 17:14:56] "POST /search HTTP/1.1" 200 -
```