Twitter Job Data Analysis Project

Project Overview

This project consists of two main components that work together to collect, analyze, and report on Twitter job market data. The system scrapes job-related tweets and generates comprehensive analytical reports with visualizations[1].

Architecture & Components

Component 1: Data Collection System

```
# Main scraping class
class TwitterScraper:
    def __init__(self, headless=False)
    def setup_driver(self, headless=False)
    def search_hashtags(self, hashtags, max_tweets=2000)
    def infinite_scroll_and_scrape(self, max_tweets)
    def extract_tweets_from_page()
    def extract_single_tweet(self, tweet_element)
    def extract_engagement_metrics(self, tweet_element)
    def save_data(self, filename, format='csv')
```

Purpose: Automated Twitter data collection
Technology Stack: Selenium WebDriver, Pandas, Chrome automation
Target Data: Job-related hashtags (#naukri, #jobs, #jobseeker, #vacancy)

Component 2: Analysis & Reporting System

```
# Main analysis class
class TwitterJobAnalysisReport:
    def __init__(self, csv_file)
    def load_and_prepare_data()
    def perform_sentiment_analysis()
    def analyze_hashtags()
    def analyze_keywords()
```

```
def analyze_engagement()
def analyze_temporal_patterns()
def create_comprehensive_visualizations()
def generate_comprehensive_report()
```

Purpose: Data analysis and report generation[1]
Technology Stack: Pandas, Matplotlib, TextBlob, NumPy
Output: Visualizations and detailed reports

Data Schema

CSV Output Structure

Column	Type	Description
Username	String	Twitter handle without @ symbol
Tweet	String	Full tweet content
Date	Date	Tweet posting date (YYYY-MM-DD)
Time	Time	Tweet posting time (HH:MM:SS)
Mentions	String	Comma-separated mentioned users
Hashtags	String	Comma-separated hashtags

Likes	Integer	Number of likes
Retweets	Integer	Number of retweets
Comments	Integer	Number of comments
Replies	Integer	Number of replies
Views	Integer	Number of views

Installation & Setup

Prerequisites

```
# Required Python packages
pip install selenium pandas webdriver-manager textblob
matplotlib seaborn numpy
```

System Requirements

- Python 3.7+
- Chrome browser installed
- Internet connection for web scraping
- Minimum 4GB RAM for processing 2000+ tweets

Usage Instructions

Step 1: Data Collection

```
# Run the scraper
python twitter_scraper.py

# Expected output:
# - twitter_job_analysis.csv (2000 tweets)
# - Console progress updates
```

Step 2: Analysis & Reporting

```
# Run the analysis
python comprehensive_analysis.py or by using
complete_pdf_generator.py
```

```
# Expected output:
# - comprehensive_twitter_analysis.png (12-panel dashboard)
# - Comprehensive_Twitter_Job_Analysis_Report.txt
Or pdf report
 #Twitter_Job_Analysis_Complete_Report.pdf
```

Analysis Modules

1. Sentiment Analysis Module

```
def perform_sentiment_analysis():
    # Uses TextBlob for sentiment scoring
    # Classifies tweets as Positive/Negative/Neutral
    # Generates sentiment distribution statistics
```

2. Hashtag Analysis Module

```
def analyze_hashtags():
    # Extracts and counts hashtag usage
    # Identifies trending job-related tags
    # Provides hashtag performance metrics
```

3. Engagement Analysis Module

```
def analyze_engagement():
    # Calculates engagement rates
    # Identifies high-performing content
    # Analyzes correlation between metrics
```

4. Temporal Analysis Module

```
def analyze_temporal_patterns():
    # Identifies peak activity hours
    # Analyzes day-of-week patterns
    # Provides optimal posting recommendations
```

Output Documentation

Visualization Dashboard (PNG)

12-Panel Comprehensive Dashboard:

1. Sentiment Distribution (Pie Chart)
2. Top 10 Hashtags (Horizontal Bar)
3. Top 10 Keywords (Horizontal Bar)
4. Average Engagement Metrics (Bar Chart)
5. Tweet Activity by Hour (Line Chart)
6. Tweet Activity by Day (Bar Chart)
7. Top 10 Most Active Users (Horizontal Bar)
8. Engagement Rate Distribution (Histogram)
9. Sentiment Score Distribution (Histogram)
10. Tweet Length Distribution (Histogram)
11. Engagement vs Views (Scatter Plot)
12. Top 5 Most Engaging Tweets (Horizontal Bar)

Text Report Structure

```
1. Executive Summary
   - Dataset overview and key metrics
2. Sentiment Analysis Insights
```

   - Emotional tone analysis
3. Hashtag Analysis & Trending Topics
   - Popular hashtags and usage patterns
4. Engagement Analysis & Performance Metrics
   - Interaction statistics and trends
5. Temporal Analysis & Activity Patterns
   - Optimal timing insights
6. Actionable Recommendations
   - Strategic optimization suggestions
7. Conclusion
   - Summary and next steps

Error Handling & Troubleshooting

Common Issues & Solutions

Issue: Datetime parsing errors
Solution: Multiple fallback parsing methods implemented
python

```
# Handles fractional seconds and various formats
try:
    self.df['datetime'] = pd.to_datetime(self.df['Date'] + ' ' +
self.df['Time'], format='mixed')
except:
    clean_time = self.df['Time'].str.replace(r'\.\d+', '',
regex=True)
    self.df['datetime'] = pd.to_datetime(self.df['Date'] + ' ' +
clean_time)
```

Issue: Missing data columns
Solution: Graceful degradation with default values

```
# Ensures numeric columns exist
for col in ['Likes', 'Retweets', 'Replies', 'Views']:
    self.df[col] = pd.to_numeric(self.df[col],
errors='coerce').fillna(0)
```

Performance Specifications

Scraping Performance

- Rate: ~10-15 tweets per scroll cycle
- Duration: 15-20 minutes for 2000 tweets
- Memory Usage: ~200-300MB during operation
- Success Rate: 95%+ with error handling

Analysis Performance

- Processing Time: 30-60 seconds for 2000 tweets
- Memory Usage: ~150-200MB during analysis
- Visualization Generation: 10-15 seconds
- Report Generation: 5-10 seconds

Configuration Options

Scraper Configuration

```
# Customizable parameters
job_hashtags = ["naukri", "jobs", "jobseeker", "vacancy"]  # Target hashtags
max_tweets = 2000  # Number of tweets to collect
headless_mode = False  # Browser visibility
scroll_attempts = 100  # Maximum scroll attempts
```

Analysis Configuration

```
# Visualization settings
figure_size = (20, 24)  # Dashboard dimensions
dpi = 300  # Image resolution
color_palette = ['#2E8B57', '#FF6B6B', '#4ECDC4']  # Chart colors
```

Future Enhancements

Planned Features

- Real-time monitoring capabilities
- Advanced NLP analysis with transformers
- Interactive dashboards with Plotly
- Database integration for historical data
- API endpoint for programmatic access