Project Media Insights

Team Members:

Darren Colby (cnet: dscolby)

Jessup Jong: (cnet: jessup)

Abstract

The prevalence of online communication platforms like YouTube has led governmental, for-profit, and non-profit organizations to conduct marketing or intelligence initiatives to better understand the media space, reach target audiences, and counter specific narratives. With this in mind, our project offers analysts an accessible data pipeline to visualize information about their own and competitor communication initiatives. The project draws from two data sources: HTTP requests for comment data and a third party api for video transcripts. Transcript data represents the contents of the video while comments represent insights into the audience. All text data goes through a preprocessing pipeline that converts text to lowercase and remove newlines, tabs, and emojis. The comments are corrected for spelling and translated to English with the help of Bing. The strings are then tokenized, lemmatized, and assigned sentiment polarities using the VADER dictionary. The data is transformed into a dashboard containing a word cloud, comments and cumulative comments time series, sentiment time series, comment cosine similarity heatmap, and a transcript similarity barplot. The time series charts include color accessible forecast graphs with periods that can be adjusted by the user. The project also includes a tool to automate uploading videos to YouTube.

Project Structure

Our project is divided into analysis, plotting, scraping, and upload functionalities, each contained in a subpackage and executable separately. The upload subpackage contains Python scripts to upload videos to YouTube as well as JSON files with API and OAuth keys that contain limited functionality and allow anyone to examine the functionality of the upload module. It also contains an upload directory with a single video to demonstrate the capability of the module. The scraping subpackage contains scripts and functions to fetch and parse comment and transcript data using HTTP requests and the youtube_transcript_api respectively. The purpose of the analysis module is to clean data from the previous step. The plotting module contains scripts and functions for creating word cloud, time series, heatmap, and bar plot visualizations, most of which use altair. Finally, the dashboard.py script loads preprocessed data, creates charts using functions from the plotting module, and displays them on a dashboard. The upload functionality is not included in the automate.sh because it assumes more rigorous credentials through OAuth. A detailed view of the project structure is shown below.

```
├── __init__.py
├── media_insights
│   ├── analysis
│   │   ├── __init__.py
│   │   ├── process_comments.py
│   │   └── __pycache__
│   │       ├── __init__.cpython-38.pyc
│   │       ├── process_comments.cpython-38.pyc
│   │       └── process_comments.cpython-39.pyc
│   ├── automate.sh
│   ├── dashboard.py
│   ├── data
│   │   ├── cleaned_comment_data.json
│   │   ├── comment_data.json
│   │   ├── dog.png
│   │   ├── glove.d2v
│   │   ├── glove.d2v.vectors.npy
│   │   ├── preprocessed_comments.json
│   │   ├── similarity_data.json
│   │   ├── transcript_data.json
│   │   └── wordcloud.png
│   ├── Makefile
│   ├── plotting
│   │   ├── __init__.py
│   │   ├── notebooks
│   │   │   └── exploration.ipynb
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-38.pyc
│   │   │   ├── ts_plots.cpython-38.pyc
│   │   │   └── word_cloud.cpython-38.pyc
│   │   ├── ts_plots.py
│   │   └── word_cloud.py
│   ├── scraping
│   │   ├── clean_comments.py
│   │   ├── comments.py
│   │   ├── __init__.py
│   │   ├── __main__.py
│   │   └── transcripts.py
│   └── upload
│
```

```
│       ├── client_secrets.json
│       ├── __init__.py
│       ├── __main__.py
│       ├── upload.sh
│       ├── upload_video.py
│       ├── upload_video.py-oauth2.json
│       └── videos
│           └── test1.mov
├── poetry.lock
├── pyproject.toml
└── README.md
```

Code Responsibilities

Jessup was responsible for writing the code to upload videos to YouTube, request and parse comments, and run the project from the shell. Darren's responsibilities were creating visualizations and the dashboard, preprocessing the comment and transcript data, and using it to forecast into the future. A breakdown of responsibilities for acquiring and analyzing text data is shown below.

Plot Type	Data Acquisition Function	Data Source	Data Storage Type	Data Analysis Type	Description	Person Responsible for Acquisition	Person Responsible for Analysis/Visualization
Word Cloud	get_request	HTTP Request	JSON	list[str]	Word cloud from comments	Jessup	Darren and Jessup
Comment Time Series	get_request	HTTP Request	JSON	pandas.DataFrame	plot_comment_ts: This loads comment data and plots the number of comments over time	Jessup	Darren
Cumulative Comment Time Series	get_request	HTTP Request	JSON	pandas.DataFrame	plot_cumsum_comment_ts: This loads comment data and plots the cumulative number of comments over time	Jessup	Darren
Sentiment Time Series	get_request	HTTP Request	JSON	pandas.DataFrame	plot_sentiment_ts: plots average VADER score per time period and forecasts future sentiment	Jessup	Darren
Comment similarity	get_request	HTTP Request	JSON	pandas.DataFrame	plot_cosine_similarity: plots heatmap of cosine similarities of comments across different videos	Jessup	Darren
Similar Competitor Videos	get_transcript_requests	YouTube Transcript API (Third Party)	list of dictionaries	pandas.DataFrame	Plot cosine similarity of videos from another (competitor) channel based on video transcripts	Jessup	Darren

Interacting with the Project

The entire program can be run with a bash script in the media_insights folder. For the entire program, please be patient as the process takes less than five minutes. If you are in a rush, fast_automate scrapes all data in real time and launches the dashboard on already pre-processed data. The dashboard therefore does not include the last histogram, but launches in less than a minute. For a full dashboard experience, please use the automate.sh or the online version here.

```
Unset
./automate.sh
./fast_automate.sh
```

The program also contains a Makefile that can be run within the media_insights folder through the following command. The “make” or “make everything” option is the slow version. For a faster version, run everything except for preprocessed comments.

```
Unset
“make” or “make everything”

“make data/comment_data.json”
“make data/clean_comment_data.json”
“make data/transcript_data.json”
“make data/preprocessed_comments.json”
```

upload_videos file can be run with the `./upload.sh` bash file or the following code:

```
Unset
python upload_video.py --file="videos/test1.mov" --title="Summer vacation in California" --description="Had fun surfing in Santa Cruz" --keywords="surfing, Santa Cruz" --category="22" --privacyStatus="public"
```

In addition, it is possible to interact with a live demo version of the dashboard by opening a web browser and navigating to https://mediainsights.streamlit.app. For best results, we recommend calling automate.sh or using the web version of the dashboard for full functionality.

Final Product

Our goal was to develop a tool that automates uploading videos to YouTube and enable users to get insights from their videos. We accomplished this goal of providing a full media kit package for a diverse range of users. In addition, we were able to make our project run from a single shell file and add a plot for the cosine similarity of comments from different videos. From exploring the data, we've pivoted from OAuth private data that only gives insights for the user's channel to publicly available API data. Providing a view count time series or a geographic heatmap that need OAuth credentials may pose a challenge for entry-level users. We have instead provided a more substantive data analysis through the cosine similarity analysis.

We believe that this focus on similarity scores provides the data necessary to power a content recommendation engine. This can be used for recommendations following the search results for a website. With our military background in mind, we first developed this program in the context of combating disinformation campaigns. When there are extremist views that a policymaker wants to counter, the automatic upload capabilities, insights into the audience members through comments, similarity information based on transcripts present a full suite of tools and information to counter an army of people uploading disinformation content online.

If one were to use this tool in a commercial context, one could easily use the tool for marketing purposes by scheduling a mass amount of videos, understanding the target audience of one's own and competitors' brands, and target advertising based on the recommendations from the cosine similarity graph. The current dashboard only allows an A/B testing of a few videos' similarity scores because we did not want to clutter the dashboard. For commercial use, a simple duplication of the same plots for different videos can extend this product to a larger dashboard that further aggregates similarity scores of channels that each contain multiple videos.