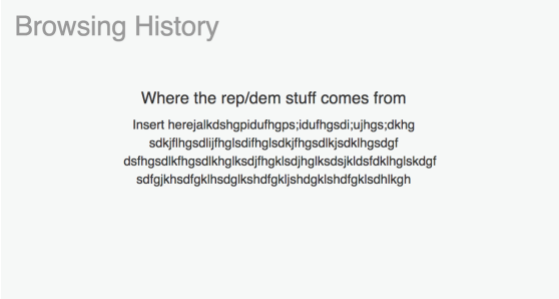
Peer Review: Political Analyzer

Introduction

Political analyzer is a sentiment analyzer for Chrome, it analyzes a webpage and generates insights regarding the political standing and overall sentiment of the page (positive or negative). This will allow users and critics alike to judge the quality of articles on the page and bring about a sociological perspective to web-surfing.

Code review

The code is very well written but, lacks documentation. The README file is more or less empty with no directions as to how one will deploy this application especially if they aren't used to using Python and Google API's. Dependency management is also lacking, which leads a brute-force like installation process. Logging is achieved using Flask-module default logger, which gives a little indication on how things run-fail as the application is used. The Chrome plugin is well made, I admire the user-interface and use of D3.js for graphs. But, there are some places in the reporting page where there's some random text which could definitely be removed.



Overall, the code will readable and it works but, the dev's could improve on documentation and packaging.

User Story Review

The user stories and the overall idea is very noble and has been executed perfectly to the word. The chrome plugin not only allows user to check the bias of the page they are on but also of the bias of Twitter and Facebook users. This is a great indicator of the overall toxicity of the content of a webpage or a user profile.

System Architecture Review

The application is built on a very simple MVC architecture. The frontend is supplied by the Chrome plugin, which talks to the Python backend using the HTTP protocol and the python backend uses MongoDB for all its database related work. The python backend also uses Google Natural Language API's to generates insights and these insights are then stored in MongoDB which allows the application to generate beautiful reports on their frontend using Bootstrap and D3.js.

Data Flow

The data flows are as follows:

- Web scraper data flow: web-page -> chrome extension (scrapes the web-page) -> backend -> generating insights using Google NLP API -> back to backend -> back to chrome plugin and updated in MongoDB.
- Twitter User Bias Workflow: chrome extension report page -> backend -> fetching data from Twitter API's -> generating insights using Google NLP API -> back to the backend -> report generated for the user on the reporting page.

Dataflow is very straightforward and logical, but I believe that this app could have been ever simpler by removing the backend and directly using the Google NLP API's to generate insights.

Error Handling

The Flask Module shines when it comes to fault tolerance, an excellent choice by dev's. The Flask Module, keeps this application running while logging each error in a readable format to the STDOUT of the backend server. But, the errors that occur on the backend are not reported to the users at all, this could be an improvement in the application.

Conclusion

A very well written and beautiful application with a noble use-case, to improve the user's user experience on the web. Places that could be improved include documentation and error handling.