Combatting Echo Chambers and Bias

Theodora Chu
Kenny Xu
Brandon Solis
Nathaniel Okun

Abstract

We plan to design and build a Google Chrome extension that helps users read more news with a different slant than they are used to. When users navigate to a news site, they can click on the extension button, which will then open up a list of suggestions for alternative reading to help them explore different perspectives on that topic. We will record browsing statistics such as the amount of time spent on each article and display that to the user on a separate website.

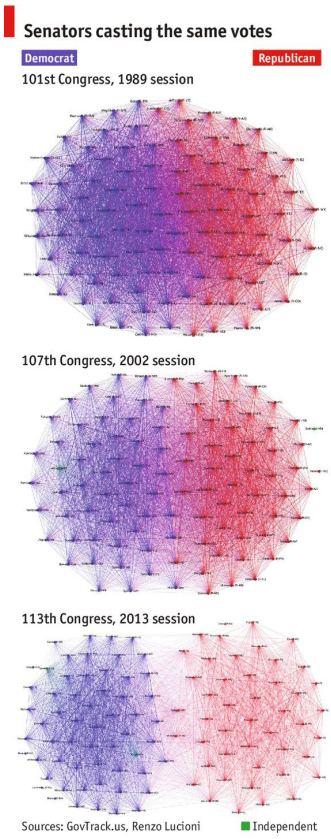
Need for Product

Increasing Exposure to Alternative Perspectives

As technology makes the world feel smaller and smaller, echo chambers are emerging more and more frequently. There is a huge demand for easily digestible and on-demand news, as is seen through the traction of companies such as TheSkimm or Twitter. However, because one of the major success metrics for these companies is customer interaction and article clicks, algorithms for news feeds become heavily skewed toward the interests or beliefs of the reader, thus creating this echo chamber where they only see news that reinforce their beliefs, which then reinforces their news feeds with more of the same kinds of opinions.

Increasing partisanship can be seen in all walks of life. Particularly evocative of this trend are Senate voting patterns (see below). In 1989, Republican and Democratic senators agreed on a variety of issues. By 2013, such an occurrence had become a rarity.

We also believe that people want to engage new perspectives. Our product will provide the momentum necessary for them to do so. “Ever since the Presidential elections, every one I know seems to be worrying about their social media echo chamber,” writes Professor Kartik Hosanagar of Wharton in an op-ed piece on Wired (“Blame the Echo Chamber on Facebook. But Blame Yourself Too”, Nov. 25th 2016). “We can easily break free but we choose not to.”



Senate Voting Patterns, 1989, 2002 and 2013
(Courtesy of Randal Olson, Senior Data Scientist at the University of Pennsylvania).

Potential Audience

This extension would target two categories of people: people who feel that current technologies and social media websites are problematically creating echo chambers and people interested in seeing all sides of a situation before forming their own opinion on a topic. These kinds of people want to be able to make well-informed opinions or judgments on events occurring in the world, so it is important for them to understand different perspectives on the same issue.

Competing Products

The main competitor is Perspecs. Perspecs is an app in the app store that shows three different articles with different perspectives for each available topic. This is meant to help users find news; however, our extension focuses less on helping users find news topics and focus more on encouraging users to read other perspectives once they have already found an article or topic of interest. We hope that the integration of our product into users' everyday lives will give it the edge over Perspecs and other competitors.

Another related but separate competitor product would be PolitEcho. This product analyzes your social media news feed to determine how biased it is in comparison to your friends' news feeds. Furthermore, it assigns each of your friends a score to determine their political bias. This is meant to help users become more aware of their echo chambers; however, again, it does not necessarily encourage users to read other perspectives nor does it actually help them in finding other perspectives.

Because awareness around echo chambers and the issue with biased news feeds only recently gained traction, few finished products out there that deal with this issue.

High-level tech design

Chrome Extension

The product is composed of two different modules - a Google Chrome extension and a server used to host, store and process news articles. The chrome extension will send the article currently being read to the server for processing. Then it will return a suggested opposing article. It will handle all front end needs and communication with the user.

Python Server

We plan to write the server in python, on top of the Django framework. We chose python because it offers many well tested machine learning and natural language processing libraries that lend themselves to the algorithmic portion of this project. We will run a script every 24 hours that scrapes articles from a list of reputable news sources. Nearly all news sources support an RSS feed - a "rich site summary" that makes it easy for programmers and readers alike to quickly access information. We will use these feeds and a python scraper designed specifically to work with RSS feeds (FeedParser, https://pypi.python.org/pypi/feedparser) to collect this information.

The final role of the server is to process each of the articles to identify their topic and other relevant information that will allow us to match it with a given article. We will preprocess the articles using the algorithms mentioned below. Doing so will allow us to expose an API with which the chrome extension can communicate and identify new articles relevant to the user.

Machine Learning Approaches

Because our project is meant to suggest articles to the user, the bulk of our long-term work will be centered around natural language processing. There are two major machine learning problems that we must tackle: discerning the topic of an article and the perspective of this article regarding this topic. All our machine learning will be done in Python, using the Sklearn package. We plan to reach out to professors for advice in improving our baseline approach, outlined below:

To predict the article topic, we plan to use Latent Dirichlet Allocation (LDA) with k-means clustering. LDA is a generative model that allows us to take any given article and compress it into a vector of finite size. To learn the topic vectors themselves, we will aggregate thousands of articles from a diverse range of news sources and run LDA on each one, producing vectors that we will then cluster with k-means. The resulting centroid vectors will be our "topics". We will need to optimize the size of each vector resulting from LDA and the number of clusters (topics). We will need a manually labelled article dataset, which can be relatively small, to use as an evaluation metric for optimization.

To predict the type of "perspective" an article has, we are going to start with a baseline sentiment analysis using a Naive Bayes classifier provided by NLTK. We will start with only two classes: "positive" and "negative" sentiment.

Putting it all together, given an article, we will label the sentiment as either "positive" or "negative". Negative articles on the same topic will be suggested for positive articles, and visa versa. In this fashion, we will have built a basic recommendation system that we can later tweak and optimize for a more refined sense of "perspective" than "good" and "bad".

The code will be hosted on the python server, so that incoming requests for article suggestions can be fulfilled.

Resource requirements

Our project requires only a server. We plan on using the free hosting services provided by Heroku.

Why a Chrome Extension?

A Chrome Extension is one of the few tools that can freely interact with users as they browse the web. We want this product to be a regular part of information gathering. Only a Chrome Extension makes this possible.

Assessment of risks

We face two main categories of risk. The first is technical risk. It may take us several tries to identify a combination of features and algorithms that accurately identify articles with the same topic but different partisan leanings. In addition, we need to mitigate the security risks that come with recording browsing information.

The second category of risks are product based. We believe that users will want to read articles with a different partisan bias and that their view of world might be changed by it. However, this may not be the case. It is possible that users may find their worldview extremely compelling and find the articles we present intrusive. Additionally, presenting readers with opposing views may in fact cause them to more solidly withdraw

into their own. To mitigate these risks, we will have to test our product early and often, and collect user feedback. Also we hope that users actively downloading our task will be more willing to accommodate different viewpoints. Fortunately, our product is relatively easy to manually reproduce. We hope to use this fact to test our product before investing large amounts of time into building technical infrastructure and adapt it accordingly.

Next steps

We need to investigate the degree of confidence we have in our model. We will build a small database of articles to be characterized manually by political leaning. We will also build a baseline for our model. We can use the articles to train and test our model. After we are satisfied we can link it to the database aggregated by our article collector and continue further testing. We also plan to meet with NLP-focused professors to get a second opinion on our algorithms and whether or not they have any recommendations for other algorithms to look into.