Congress and Me

Technical Report

Abu-Zayd Abdullah
Ben Burton
Calvin Dao
Robert Gutierrez
Ben Hazel
David Sikabwe

congressand.me

Motivation

The six of us believe that a well-informed populace is crucial to a functioning democracy. We wanted a way for people to easily see what issues their representatives are and aren't talking about in their tweets and on the Congress floor, and we wanted to highlight which issues require attention on a nation-wide scale. To that end, we've built Congress and Me.

Features

Searching

The splash page allows a site-wide search function that is split into tabs for each model: Representatives, Issues, and States. Also, you can a perform a search on each model page as well. We use a library called Fuse.js to implement fuzzy searching. All the attributes for each instance of each model have different "weights". So, for example, if I searched for "al", Alabama would show up before Alaska because the Abbreviation attribute has a higher weight than the state's name.

Sorting

A user can sort any of the three model pages either alphabetically or reverse alphabetically. When user clicks on either A-Z or Z-A in the dropdown menu, a useEffect is triggered that sorts the data (which is already loaded on the client side) either alphabetically or reverse alphabetically, gets a slice, and displays one slice—one page—at a time.

Filtering

We allow users to filter Representatives by State, filter States by Issues (for which states is a top issue?), and filter Issues by State (what are the top issues in this state?). We do this by adding a filtering parameter to the API call that sets up the endpoint we pull data from.

Pagination

To implement pagination on the model pages, we make a call to that model's API endpoint. With the data for every instance loaded up, we then divide the data into slices and display only a page's worth of data at a time. We decided to do our pagination on the front end rather than the backend to make it easier to paginate filtered search results.

User Stories (from *Quittin' Emittin'*, our Customer)

Phase 1

- *As a user, I would like the state's list of congress members to link to their specific web pages.*
- *As a user, I would like to be able to view pictures of each and every congress member.*
- *As a user, I would like an about page that gives me more information about the developers and their individual contributions.*
- *As a user, I want a homepage that includes a basic description of what the website is about.*
- *As a user, I want to be able to navigate between pages at the top of the screen.*

Phase 2

- *As a user, I would like my scroll height to be reset when changing pages.*
  - This issue kind of... fixed itself?

- *As a user, I would like an instance's picture on the model page to be clickable.*
  - This was implemented in about 15 minutes.
- *As a user, I would like a footer on the bottom of (most) the pages.*
  - We made the executive decision not to implement this issue.
- *As a user, I would like different background photos for each model pages.*
  - We implemented this in about 30 minutes.
- *As a user, I would like social media links to open in a new tab.*
  - This... also kinda just started working... on its own???

Phase 3

- *Add five attributes to States Model Page.*
  - Picture, state name, state abbreviation, summary, and website. We estimated that this would take an hour. It took an hour.
- *Even out Pagination on States Page.*
  - This took precisely 60 seconds. All we had to do was was have nine per page instead of ten. Literally just took a minute as we projected.
- *Support Filtering on all three model pages.*
  - We've added a dropdown menu to each model page. We allow users to filter Representatives by State, filter States by Issues (for which states is a top issue?), and filter Issues by State (what are the top issues in this state?). We estimated that this would take 12 hours. It took 14.
- *Support searching on all three model pages.*
  - We use a library called Fuse.js to implement fuzzy searching. All the attributes for each instance of each model have different "weights". So, for example, if I searched for "al", Alabama would show up before Alaska because the Abbreviation attribute has a higher weight than the state's name. We estimated that this would take 10 hours. It took 10 hours.
- *Support sorting on all three model pages*
  - A user can sort any of the three model pages either alphabetically or reverse alphabetically. When user clicks on either A-Z or Z-A in the dropdown menu, a useEffect is triggered that sorts the data (which is already loaded on the client side) either alphabetically or reverse alphabetically, gets a slice, and displays one slice (one page) at a time. We estimated that this would take 10 hours of coding. It took 8.

User Stories (to *The Last Word*, our Developer)

Phase 1

- *As a user, I'd like to see where a given language fits on a language cloud.*
- *As a user, I'd like to see the most spoken language in each country.*
- *As a linguist, I'd like to see the IPA transcription of a given language's autonym.*
- *As a user, I'd like to have a pretty splash page greet me upon visiting the site.*
- *As a user, I'd like to see the ISO language codes for a given language.*

Phase 2

- *As a user, I'd like a short bio about and picture of each of the developers.*
  - The about page looks a little... unfinished. It appears that you're pulling all the information from GitLab. You should ask the dev team to add a profile picture to their GitLab profiles so the about page shows their beautiful faces. Furthermore, you're all very interesting people and we'd like to know all about you! Give us a little information about yourselves!
- *As a user, I'd like the country's flag to be on its country card.*

    - The graphs on the Countries model page are very pretty. Problem is, I have no idea what they represent. I think it would be a much better idea to have the country's flag as the image. It would make countries easier to spot (my eye would see the Canadian flag a lot faster than the word Canada).
- *As a user, I'd like the links to charities to open in a new tab.*
    - Currently, the links open in the current tab. You want people to stay on your site for as long as possible (so they can see more ads 😉). We want as little interruption of the browsing experience as possible.
- *As a user, I would like the About page cards to not be clickable.*
    - Currently, the about page cards very much look like they do something or link to something. When you click on them, there is an animation, but nothing happens. They should either link to something or not be clickable.
- *As a user with 30 tabs open, I'd like a non-default favicon on the tab.*
    - Maybe put the little waving dude up there. It would help your web page appear more professional. Also, it would help the user quickly identify which tab is The Last Word.

Phase 3

- *As a user, I'd like to be able to search site-wide.*
    - I think it would be convenient to be able to search for everything on the site. Maybe put it on the homepage. We're thinking that when a user first goes to the site and doesn't really know that much about it, a site-wide search function would be a helpful tool.
- *As a user, I'd like to be able to filter languages by country.*
    - We think it would be helpful if a user could, say, get all of the languages that are commonly spoken in a certain country. Switzerland has four officially-recognized languages. On the Languages model page, a user would be able to able to select Switzerland from a drop-down menu and get all the languages spoken there.
- *As a user, I'd like to be able to sort languages alphabetically.*
    - We think you should have a drop-down menu allowing a user to choose whether the Languages model page is sorted from A to Z or from Z to A. This would allow users to more easily find what they're looking for.
- *As a user, I'd like to be able to sort countries alphabetically.*
    - We think you should have a drop-down menu allowing a user to choose whether the Countries model page is sorted from A to Z or from Z to A. This would allow users to more easily find countries they're looking for.
- *As a user, I'd like to be able to sort countries by speaker population.*
    - We think you should have a drop-down menu allowing a user to sort languages from most-spoken to least-spoken and vice-versa. That way, if a user were curious as to what the most spoken language in the world was, they could find it. Similarly, if a user wanted to find languages that are endangered or dead, they could simply sort from least-spoken to most-spoken.

APIs

Outsourced APIs

We gathered data on representatives, states, and issues from RESTful APIs exposed by three different websites. We used ProPublica to get information about members of Congress and issues being discussed. We used Sunlight Foundation to get information about states and who represent them in Congress. We used Google's Civic Information API to get more information about issues and Congress Members. We also used GitLab's API to dynamically display information about our team members' issues and commits.

Our API

Our Restful API has five endpoints that can be queried: Representatives, States, Issues, Mentions, and megaTable. We currently expose five methods: GET States, GET Issues, GET Representatives, GET Mentions, and GET megaTable.

All five of these methods take in three possible parameters: page_number, results_per_page, and q. The data is divided into pages (Representatives has 54 per page and States is 10 per page, for example), and if page_number is specified, then the API will return a page's worth of results for the given endpoint. Otherwise, all instances will be returned. A user can change the number of results per page by specifying results_per_page. q is a query parameter. A user can filter results according to certain parameters. For example, adding,

```
q={"filters":[{"name":"state","op":"==","val":"TX"}]}
```

to a GET Representatives request, gives you all representatives where the state attribute is "TX"—all representatives from Texas.

Our GET methods also allow users to specify specific entries. For example, https://api.congressand.me/api/<model>/<model_instance>/

Models

Our three models are Representatives (members of the United States House and Senate), States, and Issues (abortion, guns, immigration, et cetera). Representatives are from states and discuss issues. States have representatives and have a most-discussed issue. Issues are discussed by representatives, and they have certain states in which they are hotly discussed.

- Issues
  - Name
  - National ranking in regards to other issues being discussed
  - State ranking in regards to other issues being discussed
  - States in which it is being discussed
  - Representatives who are talking about the topics
- State
  - Name
  - Congress Members
  - Issues being discussed in state
  - Ranking of issues being discussed
  - Issues not being discussed in state that are being discussed in other States
- Congress Members
  - Name
  - Home State
  - Party Affiliation
  - Issues that they are discussing

Testing

Selenium

We used Selenium to simulate a web browser when testing our GUI in `SeleniumTests.py`. These tests assure that each web page loads properly by asserting that the right text is displayed upon loading that page.

Postman

We test our backend API in `UnitTests.postman_collection.json`. Our API has five GET requests (one for each of our endpoints: Representatives, States, Issues, Mentions, and megaTable) and we test the length, type, and number of pages of the response for each GET request.

Jest

Jest is a framework built into React for testing JavaScript code. We use it in `src/App.test.js` to make sure our front-end elements render properly.

unittest

unittest is a Python framework to test Python code. We use it in `backend/API_Tests.py` to test our API requests as well as in `SeleniumTests.py`.

Tools

React

Our front end uses React, a JavaScript library. We use Bootstrap, a CSS framework, for styling.

Database

Our database has four tables: Representatives, Issues, States, and Mentions. The attributes for the first three tables are the attributes for the corresponding models. The final table Mentions is a grotesque jerry-rigged association table linking Representatives, States, and Issues. The attributes for this table include a representative's name, their state abbreviation, and a column for every issue. Each issue attribute currently lists whether or not that issue is on that representatives website. In future phases, we will work on getting more robust data on frequencies of discussion. We have a similar table linking States and Issues cleverly called stateIssues that we use to for filtering the States and Issues model pages. We use the Flask-SQLAlchemy framework to manipulate our database with Python code (see `app.py`).

Slack

Slack is our team collaboration tool. We have different channels for different topics like front end, back end, AWS, et cetera.

GitLab

We use GitLab for version control. GitLab's issue boards are helpful for keeping track of which members of our group are assigned to what tasks as well as allowing us to connect with our customer group—they can provide issues and we can discuss the issues with them.

Flask

We use Flask, a Python web application framework, to access our database with Python code. Flask-SQLAlchemy, an extension of Flask, lets us create API endpoints in our Python code.

Docker
Docker creates a virtual environment on which to run code. Our EC2 instance uses our Docker image to run our backend.

Visualization Tools
To create our visualizations, we mainly used D3.js (a JavaScript library for data visualization) or tools that were based on or built around it. The choropleth is made from vanilla D3. The pie chart is made from a D3 reimplementation called Victory. We made the word cloud using a component called react-wordcloud.

Hosting

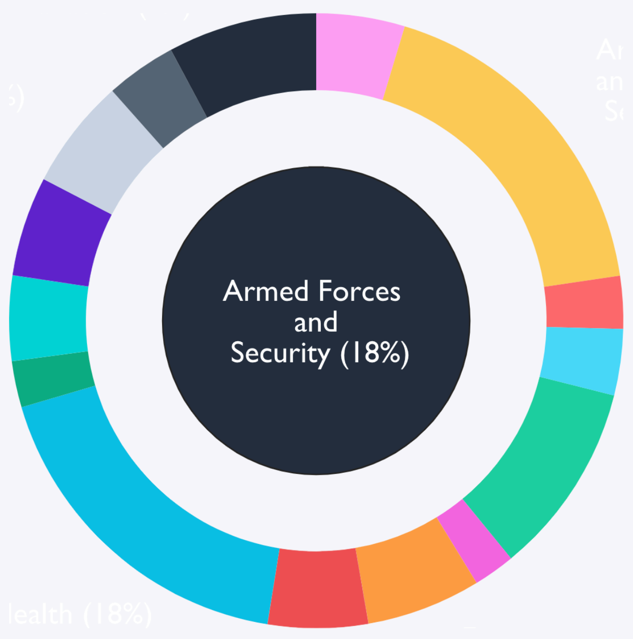
Amazon Web Services
Our front end is a React app that we store using S3, a cloud storage service within AWS. This S3 bucket is deployed through CloudFront, AWS's content delivery network. Our backend is a Flask app running on an EC2 (Elastic Cloud Compute, also an AWS service) instance. Our PostrgreSQL database is stored on AWS's Relational Database Service (RDS).

Domains
We acquired our pretty URL congressand.me from NameCheap (we used to be Congress Conversation, but Congress and Me is conducive to a free URL). This URL is used as a CNAME alias that points to our CloudFront endpoint. We obtained an SSL certificate through the AWS Certificate Manager, which allows us to secure our site with HTTPS. Our API endpoint is registered as a CNAME through Elastic Beanstalk.

Visualizations

Visualization 1

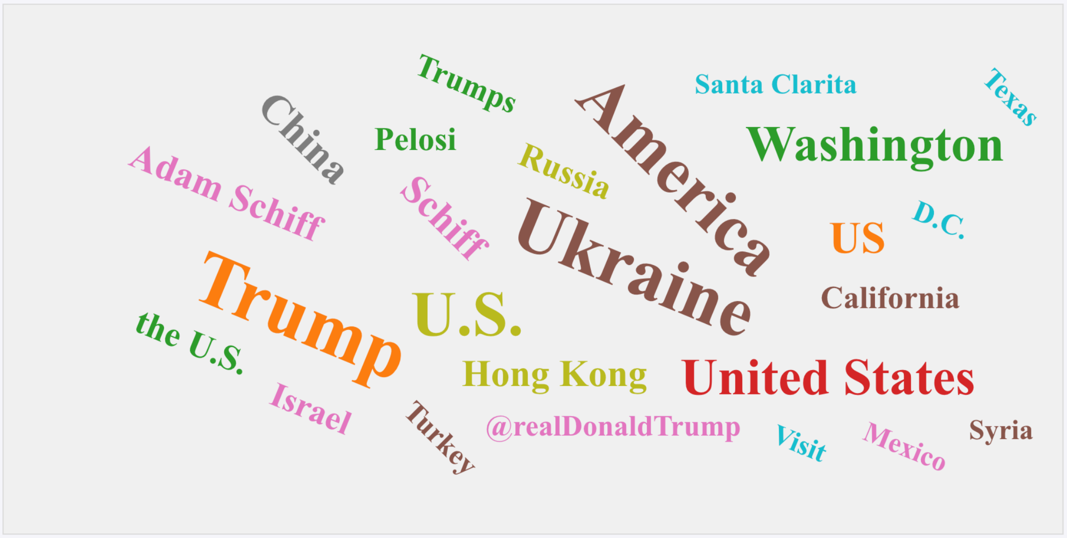
Relative importance of each issue. (Hover over a slice.)



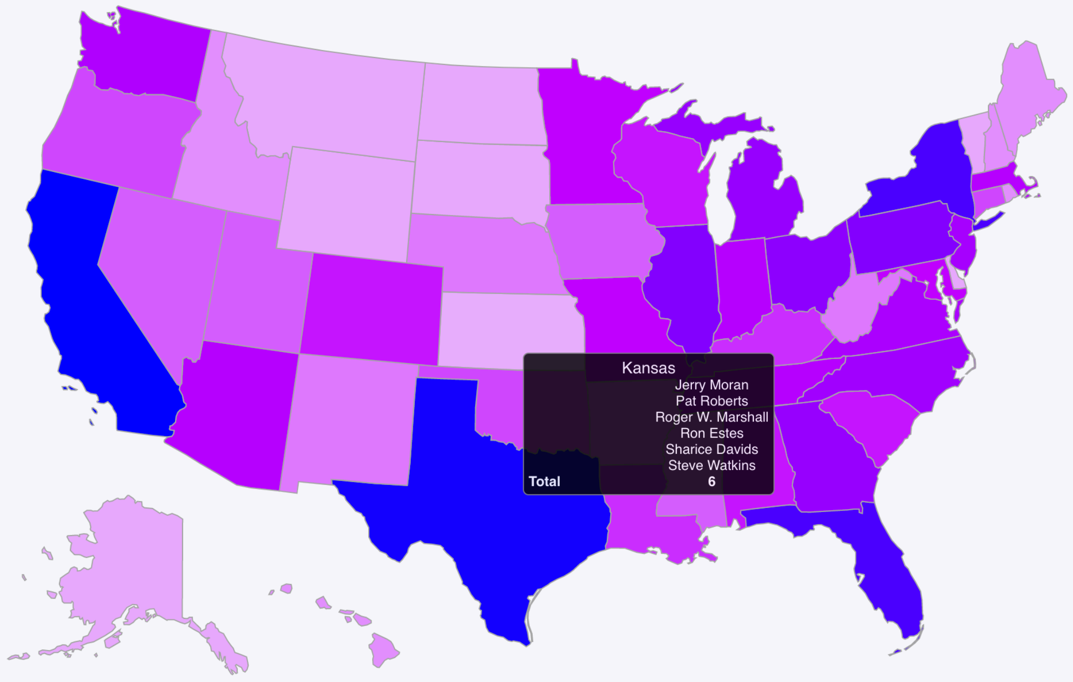
The first visualization shows the percentage (based on tweet data) of representatives who are discussing each issue.

Visualization 2

Representatives' most commonly tweeted words. (Hover over a word.)



The second visualization shows the most common words tweeted by representatives. The size of the word is indicative of its frequency in the tweet data.



The third visualization shows how many representatives are in each state and lists them upon hovering over the state with the mouse. The more lightly-colored states have fewer representatives.