# Final Project

## Narrative of application and data model

The proposed application for this dataset is a website that will let users explore topics discussed in Congress over the last year of Obama's administration and the first year of Trump's. The main page displays passages from speeches on a given topic, ranked and filtered by date, party affiliation, and state represented by the speaker. For example, if a user selects the topic "immigration" from a filter menu on the left, they're shown passages from speeches where keywords and synonyms for that topic appear. Additionally, users can limit results by setting a date range or selecting a party affiliation, a house, or a state. The documents are ranked by relevance, where statements that are out of the ordinary for their speakers would be more relevant than statements that are typical for their speakers.

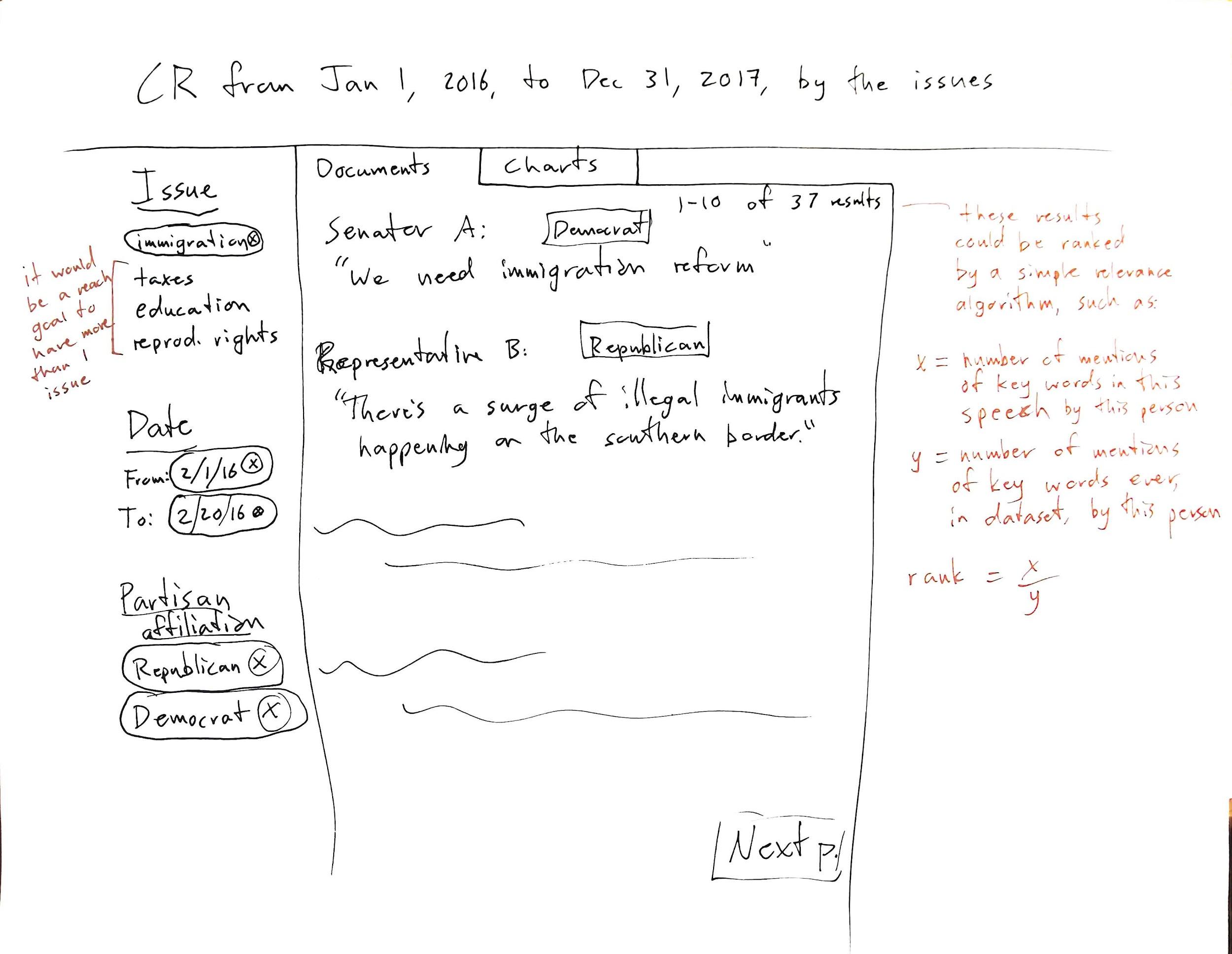


Figure 1. Wireframe of user interface.

The data model is composed of three main components, texts, people, and topics. The texts are modeled as issues of the Congressional Record, with speeches as a subclass. A speech is the text of a single speaker, however short or informal, that is recorded as part of the business of congress. Speakers are kept track of with a separate class, and topics a third top-level class. We decided to keep topic a separate class from text since multiple topics might appear in a given speech, and multiple speeches might address a single topic.

## UML Models

## XML / JSON Serializations

Our example serializations are provided in XML, with RELAX NG files to validate them. One XML file is used for each of the top classes, *crIssues*, *congressPerson*, and *topic*.

## External vocabularies

For the XML serializations, we chose to incorporate two standard external vocabularies, Dublin Core and Friend of a Friend. First we set an imaginary default namespace in each XML document (“https://example.com/maryse\_and\_joe\_project/”) with which to define all the elements that aren’t given namespaces. For metadata at the issue level, we decided to keep the MODS terms provided for the *Congressional Record* by the govinfo.gov API, since it may be important for the application to link back to the Congressional Record website. For the speeches we identified within individual issues, we decided to incorporate the basic Dublin Core vocabulary for some of the elements in *crIssues*, as DC provides an interoperable and recognizable solution to common text attributes. There is a limitation here in that these two vocabularies cover the same type of domain, so they may be confusing to be used in close proximity. Finally, we used FOAF to standardize some of the information for congresspeople, since that is the best basic ontology we know of to describe people, and the URI of a Library of Congress Subject Heading as the ID for our example topic, immigration.

## Ontology

## RDF triples

[Do we need the text below? Didn’t delete in case it might be useful.]

* The *crIssue* class is for date and publication information for the Congressional Record issue. Its attributes currently consist of: *crId*, *date*, *volume*, *number*, and *pages*. The pages attribute currently has the children *pageStart* and *pageEnd*. Multiple *crIssue* instances are wrapped with a *crIssues* element.
  + The *document* class appears as a child of *crIssue*. It is for information regarding the speech results returned from the user’s search query. Its attributes are: *docId*, *title*, *subtitle* (<zeroOrMore>), *author* (<oneOrMore>, related to the *congressPerson* class by *personId*), *crSection* (which section in *the Congressional Record* issue this document is located), *relevance*, *type* (which will always be “Speech”), and *text*. The *text* attribute is still being formatted as we consider if we want the data model to reflect the paragraphs in the original document.
* The *congressPerson* class is for information regarding the author(s) of the document class. Its attributes are: *personId, firstName*, *lastName*, *chamber*, *state*, and *partyAffiliation*. Multiple *congressPerson* instances will be wrapped with a *congressPeople* element.
* The *subject* class is to represent the topics addressed throughout *document* instances. Each *topic* will have a *subjectId* and a set of *expressions* and *documents*. Each *expression* element is defined by a regular expression string (*regEx*) and carries a *totalFreq*, the number of times that regular expression matches throughout the dataset. Each *document* element represents one document containing this topic. It links to a *docId* and has *docFreq*, the sum of all topic expression instances matching in the document, and *relevance*, the *docFreq* minus the sum of all *totalFreq* values for the topic expressions.