Cornell University

Computer Science

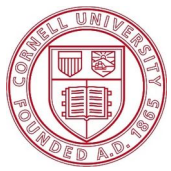
M.Eng Project Report

Super Seminar Scraper

Author:
Jiankun Lu

Supervisor:
Dr. David Bindel

December 14, 2015



Abstract

Many universities hold seminars and colloquiums every week. Although organizers usually have websites to publish their agenda, different departments may use different style and format, even in the same university. This creates difficulty when people want to gather a set of all seminars information. Super Seminar Scraper are able to fetch and parse all seminars information from different departments' website and display them in a uniform format. This report will focus on the front-end part of this system, which displays the collected and parsed seminar information in a beautiful and creative way.

1 Introduction

This data-driven website is built using JavaScript and HTML while the back-end task like data parsing are handled by Angular.js and the animation are handled by jQuery. All seminars information will be stored in a JSON file called alltalks.json and will be requested by the Angular.js $http on the user's browser. Thus there is no database or server side program involved, which made the website lightweight and easy to maintain. When a user open the website, the Angular.js controller will parse alltalks.json once it's downloaded and displayed them on the homepage. There is a map page that shows the location of hosting universities on a Google Map by using its JavaScript API. The three data visualization graphs: word cloud, donut graph and bar chart are done by using the D3.js library with the data parsed from files.

2 Implementation

2.1 The main JSON file

This website relied on the data given in the JSON file `alltalks.json` to work properly. The JSON file is located at `./data/alltalks.json`. It contains a list of records and a list of speakers, each record contains Topic, Speaker, Time, Venue, University, URL, Description and Tags. Tags is a list of keywords that extract from the abstract and/or the description of the seminar. The following is an example record entry and an example speaker entry.

```
{
records: [
  {
    Topic : Scientific Computing,
    Speaker : Jane Doe,
    Time : 2015-10-07 09:30,
    Venue : Gates 701,
    University : Cornell University,
    Url : https://cs.cornell.edu,
    Description : Lorem Ipsum...,
    Tags : [Scientific Computing, University of Arizona]
  },
  ...
}

speakers: [
  {
    Name : Paul Hovland,
    Affiliation : University of Arizona
  },
  ...
}
```

The generation of this JSON file, including the data crawling and parsing is done by my teammates. Since every departments use different format and style, the challenge is how to parse them into the uniform format as this JSON file shows and tags extraction which may need some frequency analysis.

2.2 Overall Website Framework and Style

I used bootstrap CSS to style the whole website. Some fine tuning and other CSS statements are located in `./css/main.css` or in the corresponding html files. All Angular.js controller are located in `./js` folder. There are four pages: index, map, trending and about. Each page is an html file has an identical navigation bar and a footer. The navigation bar has the link to all the pages.

Angular.js handles the logic layer of this website such as parsing the JSON file, while some small jQuery functions handle the layout such as the animation in the html files.

2.3 Help Window

In order to help the new user to familiar with the website as soon as possible, a help window (Figure 1) will pop up when the user visits the homepage.html or the map.html for the first time. I used the cookies "visited_home" and "visited_map" to judge if this is a new user or not. If the cookie has been set which means this user is not a new user, thus the help window will not pop up automatically, but the user can still click the "Help" on

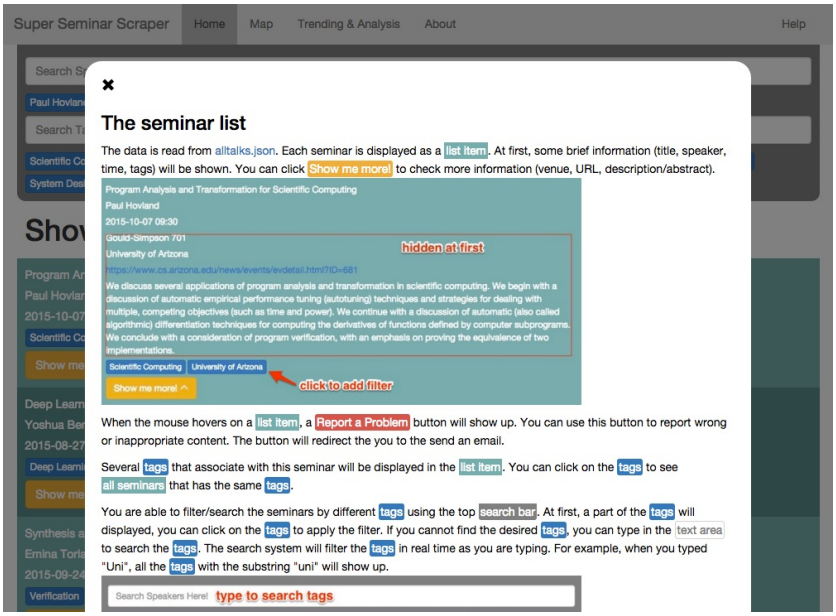


Figure 1: Homepage help window

the top-right corner to open the help window. The help window is hidden at normal time by set the CSS value `display` to none, change it to block will unhide the help window.

2.4 Homepage

This is the main page of this website and will list all the seminars information. Users can use the search function to filter seminars by tags. Once the program loaded the `alltalks.json` file, it will store all records into `$scope.talks`. `$initAllTags()` and `$initAllSpeakers()` functions will add tags and speakers in every records to `$scope.alltags` and `$scope.allspeakers` when they first appeared. By using `ng-repeat`, we can print out all the seminars records on the html. The tags for each records will also be printed in a `ng-repeat` loop. Here is some tutorial about `ng-repeat`. (http://www.w3schools.com/angular/tryit.asp?filename=try_ng_repeat_array)

Every list item has a "Show me more!" button, which will display/slide out additional information that was hidden at first when clicked. The animation is done by jQuery `slideToggle()` function. The little arrow in the button will also change direction by rotating 180 degree. There is no library function to animate the rotation of a `span`. So I use the `AnimateRotate()` which gradually change the CSS of the arrow's rotation to achieve the animation.
(http://www.w3schools.com/jquery/jquery_slide.asp)
(http://stackoverflow.com/questions/15191058/css-rotation-cross-browser-with jquery-animate)

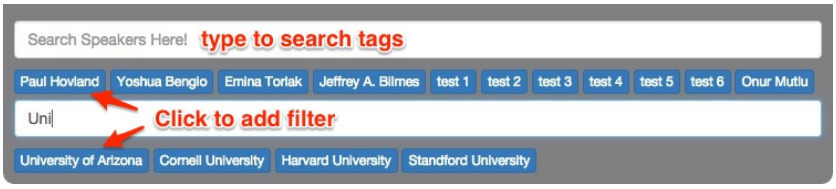


Figure 2: Search Bar

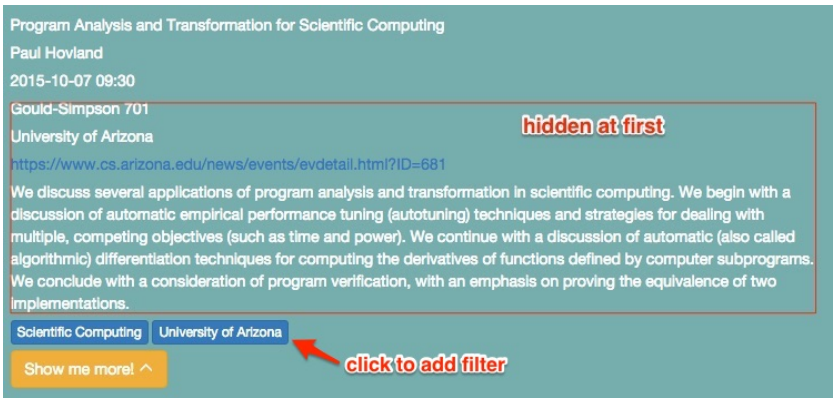


Figure 3: a sample seminar item

In the search bar (Figure 2), speakers and tags will be listed as buttons. When the user clicks a tag or a speaker button, only the seminars has that tags will be shown. Under the search bar, a show all seminars button, which will clean all filters and a list of applied filter, will also be shown. A user can click a filter button to cancel that filter. When user click a tag or a speaker button, the function `addfiler()` will be called and the tag will be add as a string to an array called `$scope.filterstring`. When user clicks to cancel the filter, it will be removed from the array. If the user click the show all seminar button, the array `$scope.filterstring` will be cleaned.

The display or not of a seminar item is determined by the `displayflag()` function that controlled by `ng-show`. Each list item will have a `ng-show` field that has a function called `displayflag(tags, speakers)` and passed in it tags and speakers as parameters. Inside the `displayflag()` function, the passed in parameter will be compared to all items in `$scope.filterstring` array. If the `$scope.filterstring` is a subset of the seminar's tags, then the function will return true and that seminar's div will be displayed. Here are some additional documentation about `ng-show` and `ng-model`.
(https://docs.angularjs.org/api/ng/directive/ngModel)
(http://www.w3schools.com/angular/tryit.asp?filename=try_ng_html_show)

If a user cannot find the tags they want, they can type in the search text area to search for the tags. The search function is achieved by the `containSearchWord()` function. The user input is bound through `ng-model` with `$searchWord`. The display of every

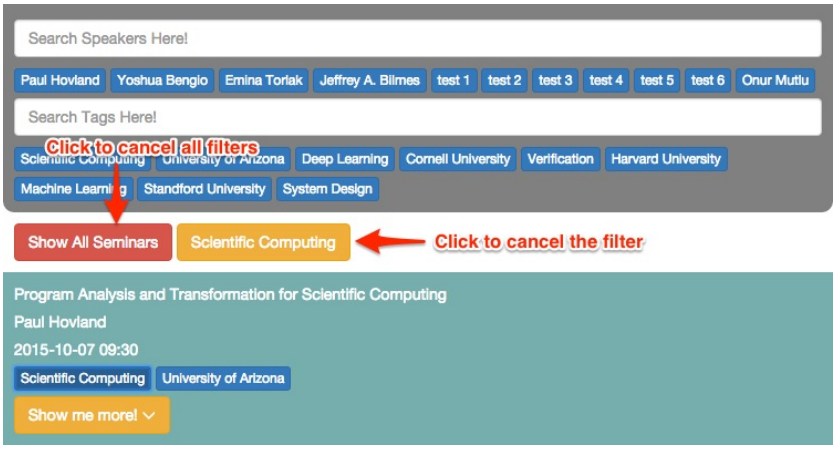


Figure 4: apply fiters

tags or speakers is controlled through `ng-show` by `containSearchWord()`. When user type a letter, the input and current tag string will be passed to `$containSearchWord()` function, where we try to find if the user input is a sub-array of the current tag string. If we can find a match, then the function will return true and that tag will be display. If there is no match, the function will return false, and that tag will be hide.

2.5 Map

When the map.html (Figure 5) is loaded, the JavaScript function `initMap()` will be called. The initiation function will load the `alltalks.json` and set the some parameters such as center latitude and longitude for the "background" Google Map.

The `loadTarget()` will loop through the records in `alltalks.json` and create a list of all universities and add relevant information to it (seminars in the same university will be put together). The location information (latitude and longitude) of universities is hard coded in JavaScript file.

Once the Map is loaded, `initMap()` will call `setMarkers()` to deploy markers on the background Google Map. I used a doctor cap picture located at `./images/cap.png` which I found online to represent different universities. The shape variable is used to define the "clickable" area of the marker. After that, the program will loop through the universities list I created in `loadTarget()` to define the marker being put on the map. Other than the markers, the program also defined `infowindows` for all universities displayed as marker, which will show the detail about university and seminars information when users click the marker. Once both are defined, a click listener will be adding to the markers that control the pop up of the `infowindow`. The Google Map API documentation has a detailed explanation.
(https://developers.google.com/maps/documentation/javascript/examples/)

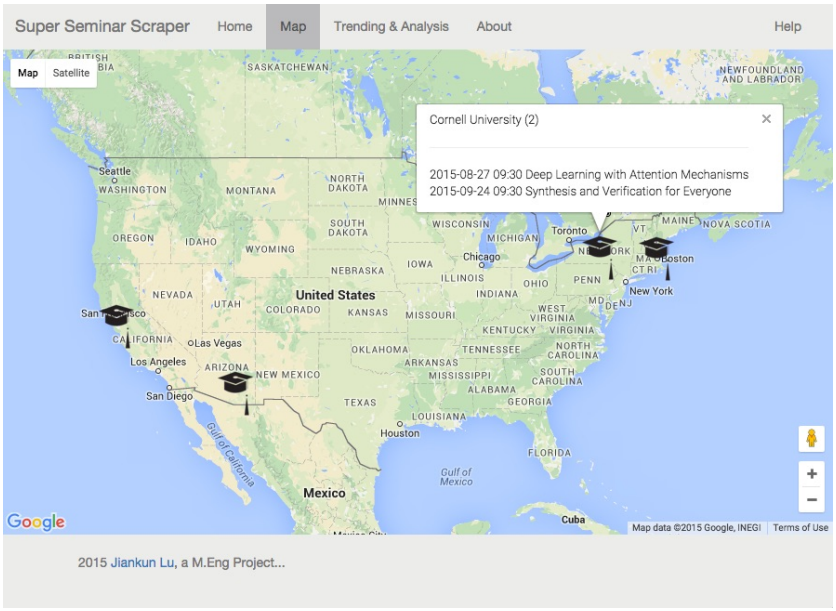


Figure 5: map.html

2.6 Trending & Anaylsis

Other than the boring information listing, the website provide another way to view the trending and analysis of the speakers and their seminars. Currently, there are three different data visualizations that show different perspective and users can interact with the first two visualizations. All of them use the `D3.js` library and the implementation is located in `.js/vis.js`.

Each visualization has a "Show Description" button that will slide out a box of a description of that visualization when clicked. This is done by jQuery `slideToggle()` function as the "Show me more!" button in the homepage. The arrow will change the direction when clicked and this is also done by `AnimateRotate()` function like the one in "Show me more!" button.

2.6.1 Word Cloud

The word cloud visualization (Figure 6) can generate a word cloud of a speaker with the description and abstract of all his/her seminars. The uesr can use a dropdown selection menu to choose the speaker. The Word Cloud function is implemented in the `wordcloudCtrl` controller using the library https://www.jasondavies.com/wordcloud that developed by Jason Davies.

This feature will use the same JSON file, `alltalks.json` as the homepage since we need to get all seminars' description, `$scope.speakers` will store all speakers information and `$scope.talks` will store all seminars information. The `select` tag in the

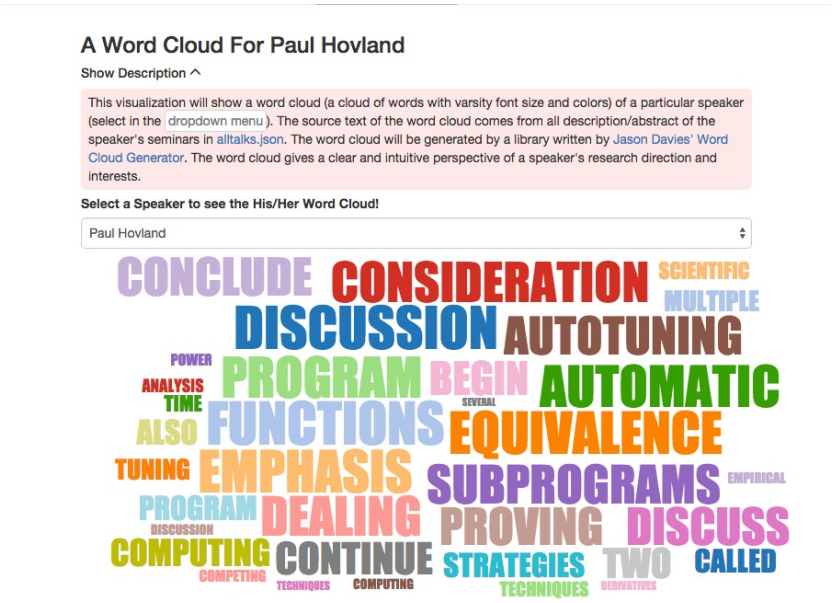


Figure 6: word cloud

html is binded with `updateWC()` using `ng-change`, so when the user select a different speaker in the `select`, it will call `updateWC()` to redraw the word cloud.

The `updateWC()` function will first use the `$scope.wcSpeaker` selected by the user to call `$scope.getWordlist()` to get a list of words to be draw on the word cloud, and then pass the word list to `$scope.drawWC()` to draw the actual word cloud.

In the `$scope.getWordlist()` function, I loop through all seminars which speaker is match the user selection and concatenate all the description together to form a corpus. Then I split the big string to a list of words and lower case all the words. To make the word list more conscious and meaningful, I excluding those words which length are smaller than three and remove all words that appears in the stop words list (a list of common pointless English words). Finally, return the filtered list for `drawWC()`.

2.6.2 Donut Graph

The donut graph (Figure 7) shows the distribution of different topic in CS seminars every months. The user can use two buttons, `[prev month]` and `[next month]` to switch the month back and forth to see the fluctuation of the distribution. I use the library code on http://bl.ocks.org/dbuezas/9306799 to generate this graph. Upon initialize, the program will read a data file called `topic_counts.json`. The first entry will list all the areas of CS seminars; this is necessary since we need this data to initialize the graph. All the records has a field called `Date` which indicated the current month of this record. We will assume that the file will be generated in a reverse chronological order.

```
{
  labels : [Cloud Computing, Security, Cryptography,
              System, Database, Vision, Graph, Theory],
  records: [
  {
    Date : 2015-5,
    Cloud Computing : 55,
    Security : 4,
    Cryptography : 20,
    System : 25,
    Database : 55,
    Vision : 77,
    Graph : 45,
    Theory : 123
  },
  {
    Date : 2015-6,
    Cloud Computing : 55,
    Security : 44,
    Cryptography : 336,
    System : 22,
    Database : 55,
    Vision : 20,
    Graph : 10,
    Theory : 120
  },
  ...
  ]
}
```

The month displayed in the title is `ng-binded` with the variable `cur_month`, so it can change when the user click the prev or next button. Once the user click the button, it will trigger the listener to increment or decrement the `$scope.pie_index` (indicate the current displaying record) and execute the `change(refreshData())` function. The `change()` function will take in a list of object looks like the following and repaint the donut graph with an animation.

```
[{label:Security; value:44},{label:System; value:50},...]
```

This input of the function `change()` will be produced by `refreshData()` function, which will loop through `response.records` and use the index `($scope.pie_index)` to determine which record need to be returned.

2.6.3 Bar Chart

The bar chart (Figure 8) tells the user how many seminars in total are taken placed in each month. Since the user doesn't need to interact with this visualization, we don't need

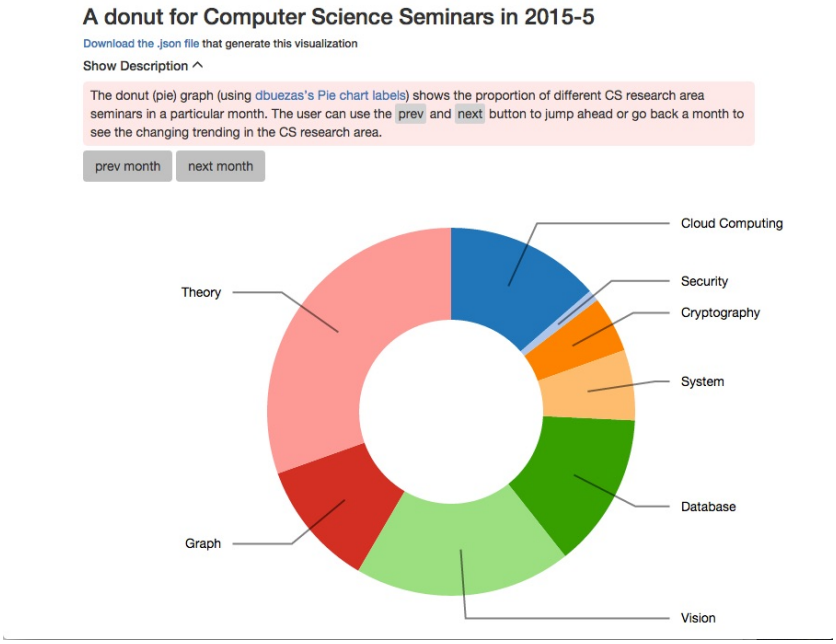


Figure 7: donut graph

to bind any variable. Thus we don't need to use Angular.js on this one. The JavaScript code for the generation of this visualization is on http://bl.ocks.org/mbostock/3885304. The program will first take in and parse a `.tsv` file which contains the data look like the following (separate by tab):

```
month count
2015-5 188
2015-6 18
2015-7 10
2015-8 111
2015-9 11
2015-11 235
2015-12 82
2015-13 53
...
```

Then the library code will append an `svg` tag under the `barchart` class and draw the bar chart using `svg` component with the data in the `.tsv` file.

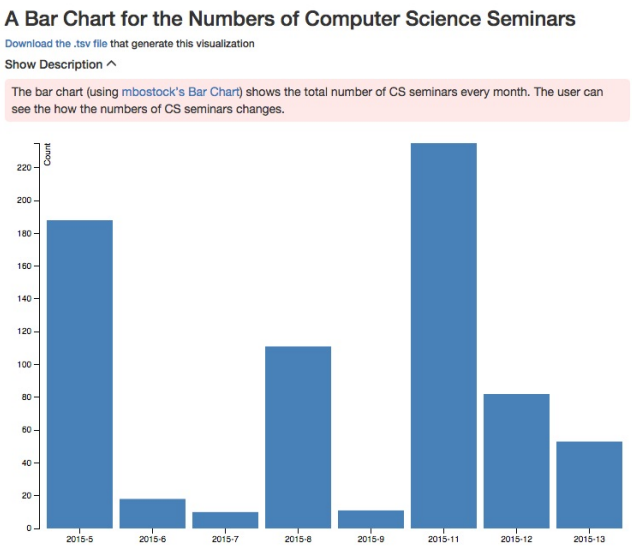


Figure 8: bar chart graph

2.7 About

This page is reserved for the description of how we build this project and other miscellaneous information

- The description about the crawl/parsing algorithm
- A list of universities website where we get the source data from
- The libraries we used in the implementation
- Source code (GitHub) links
- Contact information

The layout of about.html is shown in Figure 9.

3 Future Work

Currently, this is a working version of this tool website. But there are a lot of aspects can be improved and optimize, to make the website more usable and user-friendly.

- Right now the search function is based on tag-search. We can also add a fuzzy search so the user can still find what they want if they cannot find corresponding tags.

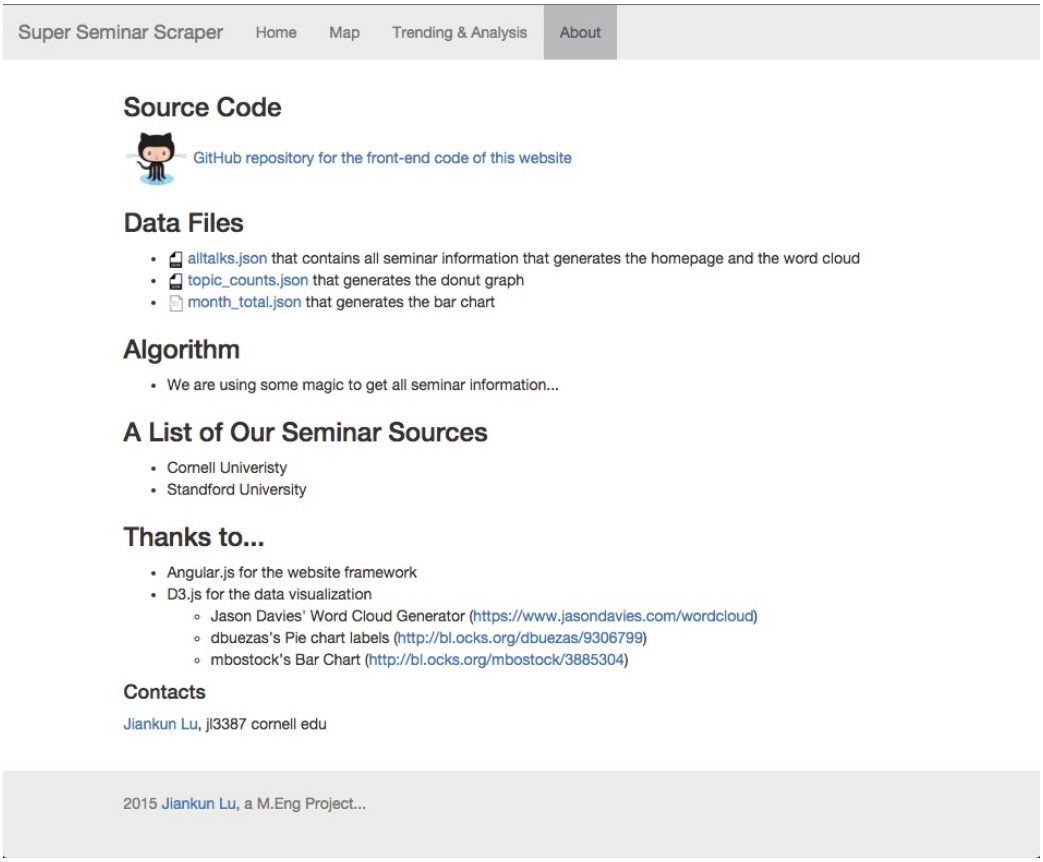


Figure 9: about.html

- Currently, the map page only show the locations of different seminars. In the future, we can add some options for user to select on the map page. For example, a user can choose to see all mathematical analysis, as a result, only the universities that hold mathematical analysis seminars will show on the map. Another example will be that a user can select a speaker, and all universities the speaker visited or are going to visit will be shown on the map.
- When zoom out the map, some neighboring markers will overlap, we can add a functionality that merge two markers when they overlapped.
- Other a static cap picture marker on the map, we can add a small red badge on its upper right to show the number of seminars in that university or that area.
- The current word cloud get token algorithm is trivial, we can implement some advanced algorithm to make the word cloud more meaningful and more accurate.
- More interesting data visualizations can be added to the trending&analysis page, using or without using existing libraries.

4 Conclusion

I mainly responsible for the front-end part of this M.Eng project. The purpose of the project is to integrate and demonstrate all the seminars and the trending in the CS academic circles. A beautifully designed website and accurate information could make people interested in this project. For myself, this is my first try for Angular.js, and I found it very handy and useful. I make use of my web development knowledge to make this website as better as I can. I hope more and more scholars can use this interesting online tool to track tendency trending in the CS academic field.