Experiment Report

Group 4

Abstract

The website is named as "Academic and More". We did our utmost to optimal the performance and beautify our website, making it both clean and informative.

Keywords: MVC, Elasiticsearch, Recommendation, Visualization

Contents

1. Back-end Integration

1.1. MVC Framework

As is known, it's sometimes troublesome to integrate components as a whole in a team-development environment.

The biggest problem lies in the transmission of data between the front end and back end. In the traditional framework, the user-interface designers have to take the data transmission into consideration. And the back end programmers also need to think how the front end will deal with these data. Besides, the high coupling traditional framework makes it difficult to modify, which clearly adds complexity to the development progress.

Hence, we need some way to separate the whole project into parts and integrate them with a simpler logic.

So we finally chose the ultra-lightweight PHP development framework Codeigniter, which provides an easy-to-use MVC structure, to build our back end structure of the website.

1.1.1. Controller

We create only one controller "Home" to handle all the pages. The methods to control the page logic are listed in this controller.

1.1.2. Model

We overall create 3 models to realize the data transmission from our database to the front end – "Author_model", "Conference_model", "Paper_model" and "Tag_model".

The first 3 models aim to deal with the data of different types of query the user execute. And the last one "Tag_model" will be use to deal with the data about the tagging system, which will be introduced later.

1.1.3. View

This part will be left in the section "Interaction between Back-end and Front-end", as well as the specific methods in the Controller and Model.

1.2. Elasticsearch

Since this project is based on the previous website, which relies on MySQL query sentence to perform searches, we found it difficult for our database to assume the demand for speed. When the sentences' structure went complicated, such as the one for collecting the data for the force-directed graph, it sometimes took almost 5s to load the simple graph. That was something hard to stand. A good website won't let its user wait that long. Even if we reconstructed the SQL sentence, the speed still couldn't meet the need of

our website. We then realized that to improve our website more radically, the former search method must be abandoned.

At that moment we had to make a choice: Elasticsearch and Solr. These two search servers are both based on Lucene, a high-performance information retrieval toolkit.

In the end we chose Elasticsearch, the newly-developing but promising one. Although Solr is more mature and provides more functions and supports more data formats excluding JSON, Elasticsearch has its advantage in usability. It is more lightweight, and provides plenty of restful APIs such as PHP and Python. This will bring great convenience for our tasks.

1.2.1. Configuration of Java Environment

Since Elasticsearch is developed with Java, it's necessary to configure the Java environment. So in the beginning we downloaded the Java Developer's Kit and did some related configuration.

1.2.2. Deployment of Elasticsearch

The deployment of Elasticsearch is also easy.

By simply running the batch file, Elasticsearch (version 6.2.4) will be deployed in the computer.

1.2.3. Installment of elasticsearch-head & Kibana

Although we have already installed Elasticsearch, we still can't check the status of our server in an easy way. Besides, we sometimes need to do some test query.

At this point the plug-in elasticsearch-head is really useful. It shows clearly every index's distributed shards and replicas, as well as the overall health status. We can also check the initial data of every index.

The installment is via git and npm.

Moreover, we install Kibana, an open-source platform for analysis and visualization. We at first expected to use it to realize part of our visualization tasks. But unfortunately, limited by time, we couldn't gain insight to the function. We simply use it as a tool to design our query DSL. This part will be left in our reflection.

1.2.4. Data Import

Given that the scale of our data is quite small, we decided to directly use Elasticsearch as our back end database, omitting the steps of synchronizing data between Elasticsearch and MySQL. So we need to import our data into Elasticsearch, which was performed with the help of Python API.

Since the database of Elasticsearch is document-oriented, which is totally different from MySQL, we need to reconstruct our data's format.

Author. At first, for every author's information, we only stored his ID, name, number of papers and the affiliation he publishes his papers most. But as we started to design the visualization part, where the data that belongs to different types are organized closely, we found out that the structure of documents was too simple to meet the demand. So we chose to create some redundancy, which is also called "denormalizing", to increase the speed of data retrieval.

For our tree view and force-directed graph, we added the information of the ID of the author's students and teachers. In this case, we don't need to execute extra query when doing these visualization.

Paper. The same reason for expanding the data storage goes for documents of papers, too. To create the several layers of the search results, we added the information of the sequence, id, name of each of its authors. And in the end, every paper's publish year and the tags are included to realize the tendency chart and the tagging system respectively.

Conference. Even though the data scale of conference is very small, we still create an index to store conference's information to further improve the performance. The information includes the conference's id, name, website, brief introduction as well as the numbers of its papers in last 10 years.

1.3. Difficulties & Improvement

1.3.1. Data Structure

It's quite regrettable that we use a very simple structure to manage our data. The indexes are isolated, with no high-level relationship to connect them. So we use a lot of denormalization to meet the need of query. This makes it difficult to design our DSL sentence, and eats up more space.

After some reflection, we find that maybe the so-called "Parent-Child" relationship can help us.

"Parent-Child" relationship allows us to deal with one-to-many relationship. We can connect two kinds of documents with a mapping and define the detailed relationship. And since they are related to each other, it would bring great convenience for us when maintaining the data and executing the query.

1.3.2. Relational & Document-oriented

When using Elasticsearch's API to execute query, we find it a bit confusing to deal with the document-oriented database. It truly differs from the relational database like MySQL. Document-oriented database aims to scale the data horizontally, while relational database tries to optimize the structure by scaling the data vertically.

It's hard to judge which one is better, since they both have their own advantages. It depends on the circumstances. But the popular solution is to use them cooperatively.

2. Front-end Development

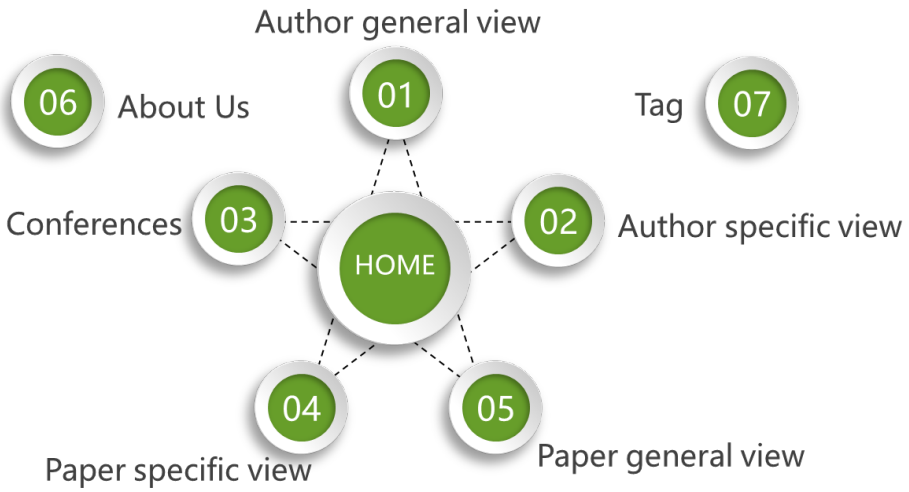
The value of a website lies in what value it can provide to its users. It depends on what the website can do, and whether it is visually acceptable for users. Therefore, as a simple paper-search product, our website use beautification and visualization to show the search result, trying our best to cater for user experience.

2.1. Beautification

As the back end and front end are connected, the websites can realize the fundamental options, but the appearance is far from beautiful. We intend to offer users the best experience of using our product. We made efforts in the beautification work, hoping to make academic search more flexible and convenient.

2.1.1. Basic Structure

The basic structure of our web page is as follows:



2.1.2. Preparation

We used some ready-made CSS files in our web designing to ease our burden and make more beautiful pages.

Bootstrap. Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Web designers can prototype their ideas or build their entire app with the Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery

We used this css file to form the frame of our websites, including the navigation bars, page layout, and the style of displaying our searched data.

Button. This module contains many different styles of button, we performed it to design our own buttons.

2.1.3. Specific Implement

Home Page.

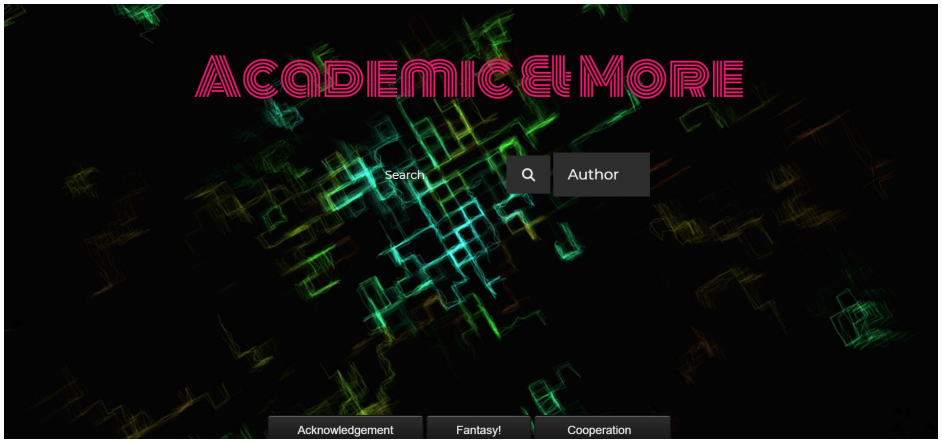
Description We set a "javascript" auto-motivated graph as our background. We added the mouse-on effect to our project name "A&M". Below is our three linked buttons, with "acknowledgement" links to "acemap.sjtu.edu.cn", "cooperation" links to "IEEE class index" and the other one "fantasy" links to three general graphs. Between them is our searching bar. Users can change their searching objects via choosing different options in the drop-down menu beside. By triggering the "search" button, they can visit corresponding pages.

Solution Design In this page, the core is to achieve the function of the conversion of search content. As is mentioned above, we add an option button after the search box, through which users can freely switch searching contents. Below is the source code.

```
<li id="options">
    <a id="change">Author</a>
        <ul class="subnav">
            <li> <a id="authortype">Author</a></li>
            <li> <a id="papertype">Paper</a></li>
            <li> <a id="conferencetype">Conference<
                /a></li>
        </ul>
</li>
```

We used the attribute "li" in the "flatnav.css". As users choose different searching options, data is sent back to the backend via different ids.

Effect Display Our home page looks like:



Result Page of Scholar.

Description This page shows ten of the searching results. You can visit our home page and "about us" page easily through the top navigation bar. We highlight the text you are searching for and show you the number of searching results in the front of this page, you can view more results by triggering the "next" and "previous" button below. For more specific scholar information, just click on the "play" button at the end of the scholar.

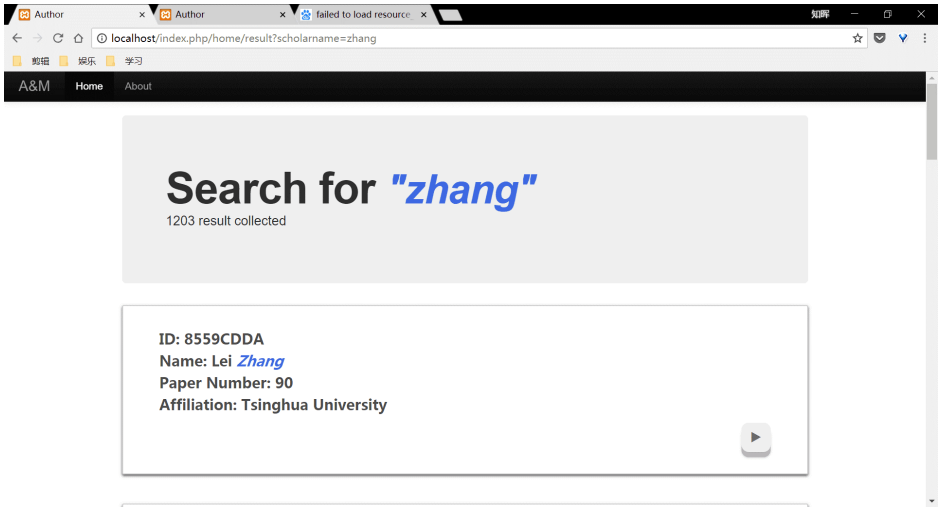
Solution Design We used models in "bootstrap.css" to design the Navigation bar. Below is the code.

```
1     <div class="navbar navbar-inverse navbar-fixed-
          top">
2     <div class="navbar-inner">
3     <div class="container-fluid">
4     <button type="button" class="btn btn-navbar"
          data-toggle="collapse" data-target=".nav-
          collapse">
5     <span class="icon-bar"></span>
6     <span class="icon-bar"></span>
7     <span class="icon-bar"></span>
8     </button>
9     <a class="brand" href="/index.php/home">A&M</a>
10    <div class="nav-collapse collapse">
11    <ul class="nav">
12    <li class="active"><a href="/index.php/home">
          Home</a></li>
```

```
13      <li><a href="/index.php/home/about">About</a></
            li>
14      </ul>
15      </div>
16      </div>
17      </div>
18      </div>
```

This ready-made class “navbar navbar-inverse navbar-fixed-top” in “boot-syrap” achieve the function of top navigation bar. It links to our home page and “About Us” page.

Effect Display The page looks like:



Scholar Page.

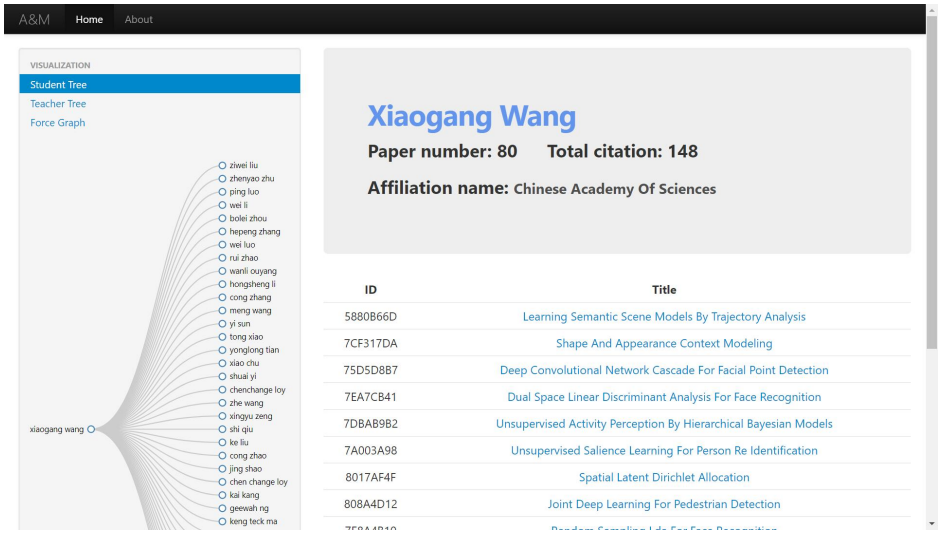
Description This page is divided into two parts, the left of which shows the teacher-student tree and the force graph. The right part displays the highlighted scholar name, paper number, total citation, affiliation name and some of his or her published paper titles.

Solution Design As we drew the visualization graphs in other pages, we need to project them into this page. We used ”Html” attribute ”iframe” to achieve it.

```
1       <iframe src="author_student_tree?id=<?php echo
            $id; ?>" frameborder='0' width='400'
            scrolling='No' height='800' leftmargin='0'
            topmargin='0' id='graph'></iframe>
```

The "iframe" tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document. This 'iframe' links to the graph in the "author_student_tree.php" identified by its 'id'. The rest of the code sets the image format.

Effect Display It will look like:



Result Page of Paper.

Description The structure of this page is the same as the previous one. The left part shows the trend of frequency of the text you are searching for as part of a paper title. But one thing we would like to mention is that we changed the way of result displaying. They are shown in the drop-down style. Every result includes three parts, the first of which is the paper title shown on the page. The other two parts are hidden unless you click on the title. As you click on it, you can see the co-author information and a "play" button for more specific information. Besides, we set tags for some of the hot words, which will be elaborated later.

Solution Design

- Accordion Pull-Down Menu

  The so-called accordion pull-down menu can display the Intrinsic connection and structure of data clearly.

```
#accordion .panel{
    border: none;
    box-shadow: none;
    border-radius: 0;
    margin: 0 0 15px 10px;
}
```

  This defines the basic form of the accordion pull-down panel.

```
#accordion .panel-title a:after,
  #accordion .panel-title a.collapsed:after{
    content: "\f107";
    font-family: fontawesome;
    width: 55px;
    height: 55px;
    line-height: 55px;
    border-radius: 50%;
    background: #ebb710;
    font-size: 25px;
    color: #fff;
    text-align: center;
    border: 1px solid transparent;
    box-shadow: 0 3px 10px rgba(0, 0, 0,
       0.58);
    position: absolute;
    top: -5px;
    left: -20px;
    transition: all 0.3s ease 0s;
  }
```

  This defines the form of panel-title after one click. The background color changes from white to yellow and the transition time lasts 0.3 second.

- Data Loop Output

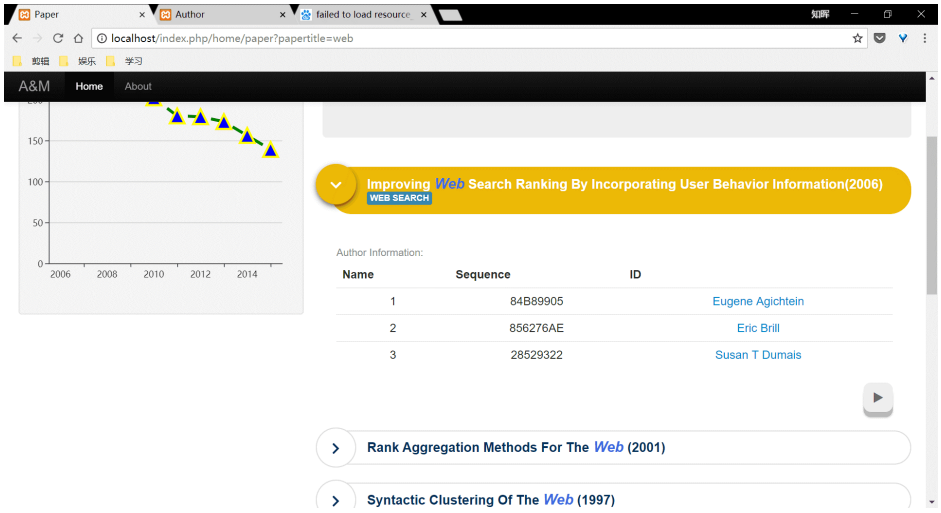
As we have designed the framework of this page, the major problem we are facing is the output of data with the specified format. Initially, we repeated the codes of creating the accordion pull-down menu ten times with different ids, this method did show ten of the results per page, but if the number results in this page is less than ten, then some empty menus appear on the screen, which is quite ugly. We solved it by adding the codes of this format into the loop-output 'php' function.

```
1       function getdata(){
2       $("#accordion").empty();
3       $.getJSON("paper_data", {'papertitle':'<?php
            echo $_GET['papertitle']; ?>', 'page':
           page}, function(data){
4       $.each(data, function(i, paper){
5       var panel;
6       if (paper['tags'].length == 0)
7       {panel = "<!-- Some codes -->";}
8
9       else {
10      var tag_content = "";
11      for (var j in paper['tags']) {
12      tag_content += "<!-- Some codes -->";}
13      panel = "<!-- Some codes -->";}
14
15      $.each(paper["authorinfo"], function(j,
           author){
16      var tr = "<!-- Some codes -->";
17      panel += tr;
18      });
19
20      var end = "<!-- Some codes -->";
21      panel += end;
22
23      $("#accordion").append(panel);
24
25      $("#b" + i + "").click(function(){
26      $(window).attr('location', "paper_ind?id=" +
            paper["id"]);});});});})
27      $("#text").html(page + '/' + pagenum);}
```

This part shows how we achieved the function of transferring the data into the panel title. As you can see, we took the method of id matching.

As we number the ids of both the data and the panel title, we can pass specific data to specific panel titles and print them on the screen. And so is the panel body.

Effect Display The page is like:



Paper Page.

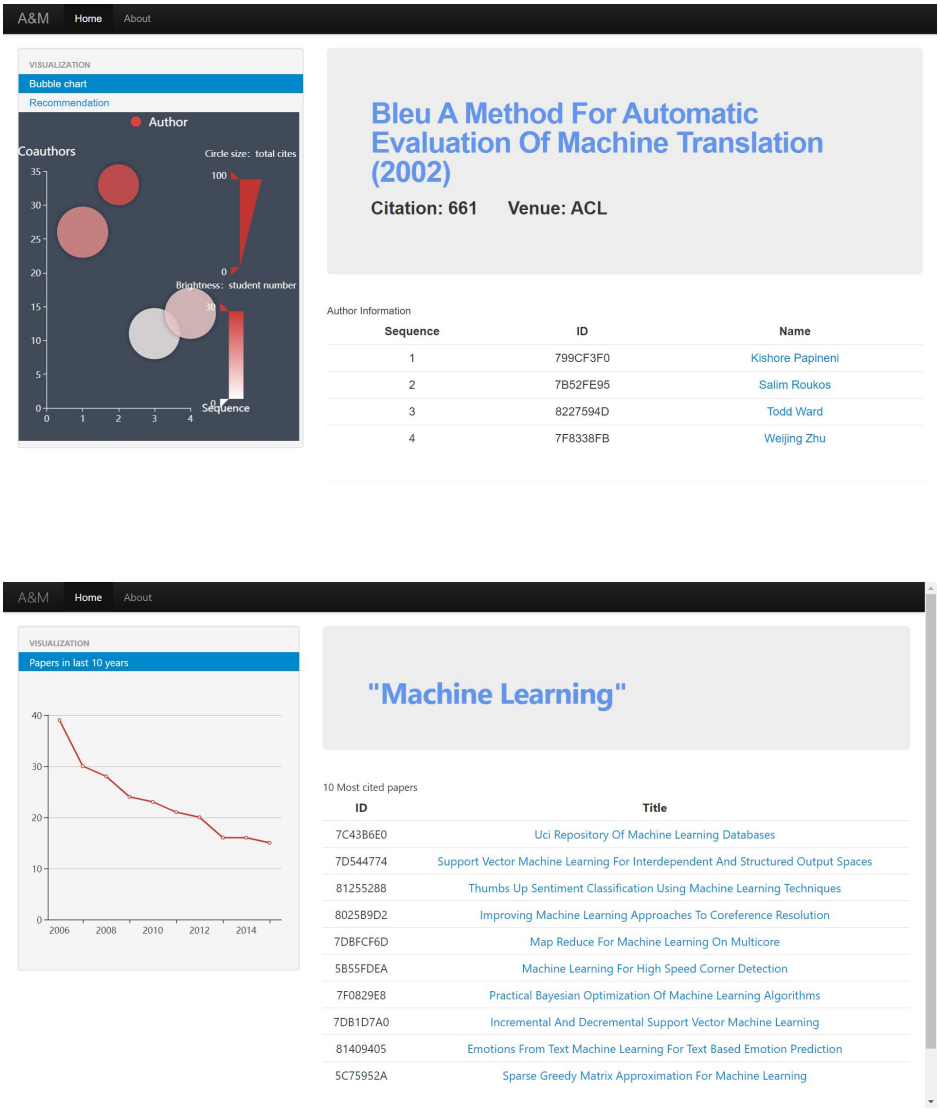
Description The page structure is the same as the previous one. However, users can visit the tag page and see the paper recommendation in this page. As you click on the tag, you can visit its page. Similarly, as you click on the text "Maybe you will like", you can see the recommended paper titles. We recommended papers from papers with the same first author and papers with the same tags.

Effect Display The page is like:

Tag Page.

Description We set tags according to popularity of common appeared words or phrases. As you click on it in the specific paper page, you can visit the tag page to derive more information. The tag page is still divided into two parts. The graph in the left part shows the popularity and tendency of this tag in recent years. The right part shows top ten most cited papers with the same tag.

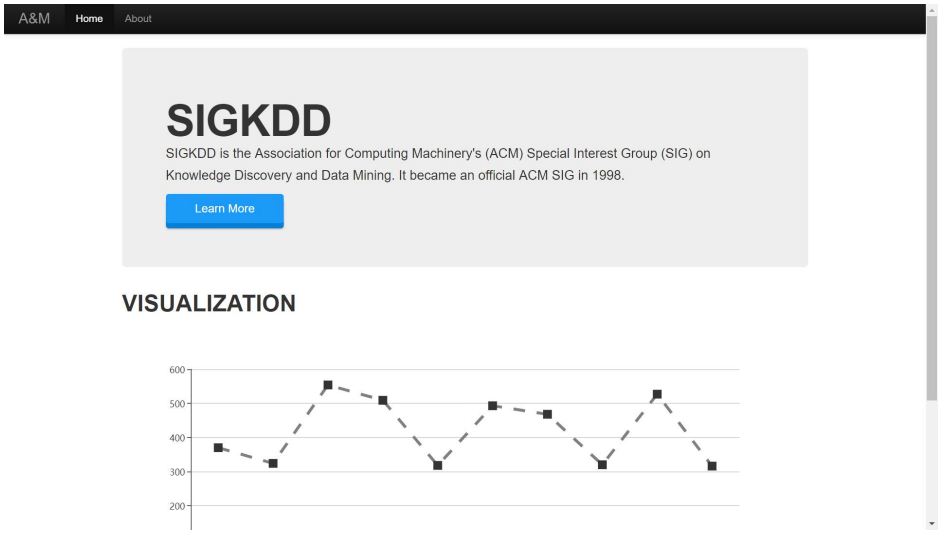
Effect Display The page is like:



Conference Page.

Description The above part of this page shows the conference name and the abstract of it. The ”learn more” button links to its official page. The graph below displays the number of papers it published in recent ten years (2006-2015).

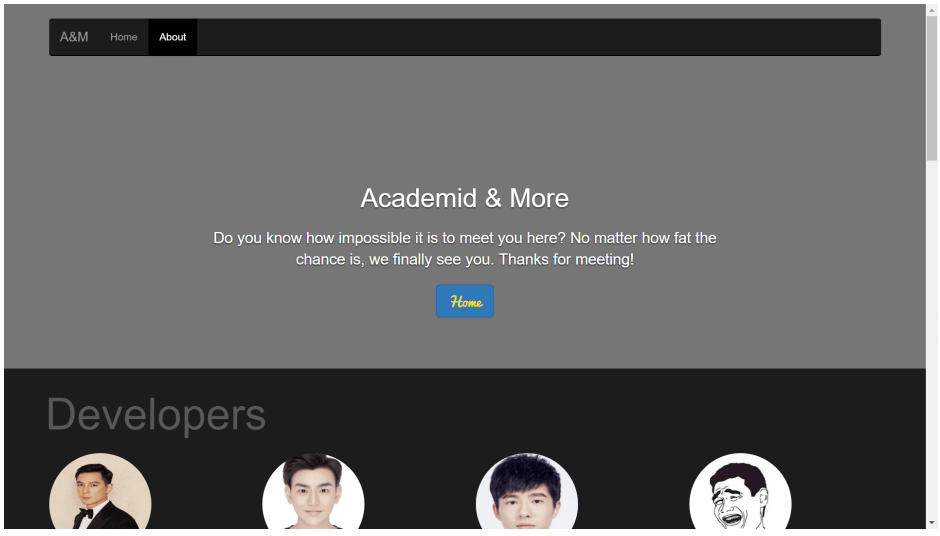
Effect Display The page looks like:



"About" Page.

Description We used "Bootsrap" framework in this page. This page shows our project name slogan and general function of this program.

Effect Display The page looks like:



2.2. Visualization

Data visualization is a scientific and technological research on the visual representation of data. Among them, the visual manifestation of such data

is defined as a type of information extracted in a summary form, including various attributes and variables of the corresponding information unit.

It is a concept that is constantly evolving and its boundaries are constantly expanding. Mainly referring to technologically advanced technical methods that allow visualization of data through expression, modeling, and display of stereo, surface, attributes, and animation using graphics, image processing, computer vision, and user interfaces. Explanation. Compared with special technical methods such as stereo modeling, the technical methods covered by data visualization are much broader.

Data visualization technology includes the following basic concepts:

- Data Space

  A multidimensional information space composed of data sets composed of n-dimensional attributes and m elements;

- Data Development

  It refers to the use of certain algorithms and tools for quantitative data deduction and calculation;

- Data Analysis It refers to the multi-dimensional data slice, block, rotate and other actions to analyze the data, so that the data can be observed from multiple angles;

- Data Visualization It refers to the processing of data in a large data set as a graphical image, and the use of data analysis and development tools to discover unknown information.

Many methods have been proposed for data visualization. These methods can be divided into a geometry-based technology, a pixel-oriented technology, an icon-based technology, a hierarchy-based technology, an image-based technology, a distributed technology, and the like, depending on the principle of its visualization.

As a paper search website, visualization is indispensable, for the vivid image we use visualization technology to draw can make our search result more intuitive, and sometimes we can find information from these graphs as well.

2.2.1. Different Graphs and Their Respective Roles

Chart & Pie Chart & Line Chart. I put these three kinds of graphs together because they share a similar role: describing the overall situation. Take pie chart as an example, we use pie chart to show thirteen conferences and they are divided into two groups according to the number of papers. In

the graph we show the exact number of papers and ratio. In this way we have an impression of the influence of each conference. Similarly, bar chart is used to show the top 10 authors with the largest number of papers and line charts are used to show the information of each year. Since they are not so detailed, mostly they are put in our home page instead of result page.

Bubble Chart. The bubble chart is put in the specific paper page and detailed information of every author is shown in the bubble, including author sequence, coauthor numbers, student numbers, name and so on. Through the bubble chart we make the connection between the paper and author closer and we can infer which author is more famous in this paper.

Map. This is a special graph in which each point represents a different affiliation, which shows its geographical position in a world map. You can learn more about each affiliation and some geography as well.

Force-Directed Graph & Tree. These graphs describe the specific relationship between the author and coauthors. Two trees display the teacher and student relationship directly, while the Force Graph displays every cooperator and their partnership clearly. It is a complement of author information and maybe you can find someone you are interested.

2.2.2. Specific Implement

Echarts. ECharts is an open-source visualization library implemented by JavaScript. ECharts provides regular line charts, histograms, scatter plots, pie charts, K-line charts, box plots for statistics, maps, heat maps, and line graphs for visualization of geographic data for visualization of relational data Diagrams, tree maps, sunbursts, parallel coordinates for multidimensional data visualization, funnel diagrams for BI, dashboards, and support for mashups between diagrams. In addition to the already built-in charts that contain rich functionality, ECharts also provides a custom series that only needs to pass in a render item function to map from data to whatever you want, and what's even better is that these are already Some interactive components work together without worrying about other things. We can download the build file that contains all the charts on the download interface. If you only need one or two of the charts and the build file containing all the charts is too large, you can customize the build by selecting the desired chart type in the online build.

Actually, some graphs are inspired by the source code from ECharts, including bar chart, pie chart, line chart and bubble chart. I will introduce them one by one.

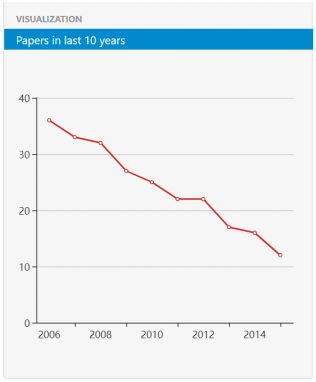
Line Chart First, we introduce external web page to import tools about ECharts, so that we do not need to set specific style and implement function for drawing.

Then we set axis by determining type and data. Later we have to choose our favorite style to decide the color and symbol:

```
option = {
    xAxis: {
        type: 'category',
        data: ['2006', '2007', '2008', '2009',
            '2010', '2011', '2012', '2013',
            '2014', '2015']
    },
    yAxis: {
        type: 'value'
    },
    series: [{
        data: [],
        type: 'line',
        symbol: 'triangle',
        symbolSize: 20,
        lineStyle: {
            normal: {
                color: 'green',
                width: 4,
                type: 'dashed'
            }
        },
        itemStyle: {
            normal: {
                borderWidth: 3,
                borderColor: 'yellow',
                color: 'blue'
            }
        }
    }]
};
```

After receiving data from back-end, a line chart is finished!

Bar Chart Similarly, we firstly introduce external web page to import tools, after which we set axis and style. While in this graph, we try to use



some emphasis and make color with gradients, so the option part is a little longer:

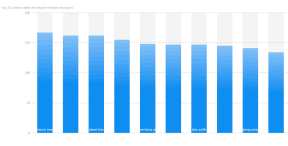
```
option = {
    title: {
        text: 'Authors',
        subtext: 'Top 10 authors with the
            largest number of papers'
    },
    xAxis: {
        data: dataAxis,
        axisLabel: {
            inside: true,
            textStyle: {
                color: '#fff'
            }
        },
        axisTick: {
            show: false
        },
        axisLine: {
            show: false
        },
        z: 10
    },
    yAxis: {
        axisLine: {
            show: false
        },
        axisTick: {
            show: false
```

```
            },
            axisLabel: {
                textStyle: {
                    color: '#999'
                }
            }
        },
        dataZoom: [
            {
                type: 'inside'
            }
        ],
        <!-- Other Options -->
    };
```

And this time we can extract data beforehand and get the graph directly. To make our graph different, we can add some functions, for example:

```
    myChart.on('click', function (params) {
        console.log(dataAxis[Math.max(params.
            dataIndex - zoomSize / 2, 0)]);
        myChart.dispatchAction({
            type: 'dataZoom',
            startValue: dataAxis[Math.max(params.
                dataIndex - zoomSize / 2, 0)],
            endValue: dataAxis[Math.min(params.
                dataIndex + zoomSize / 2, data.length
                - 1)]
        });
    });;
```

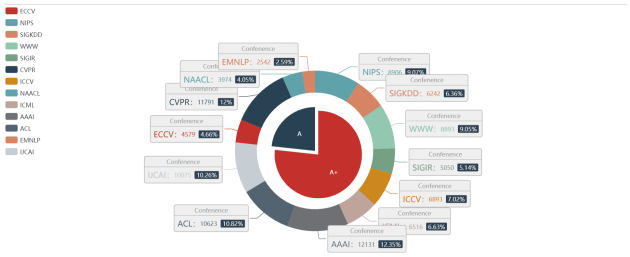
So, if we click on one of the bar, the graph will become bigger and more detailed according to the place you click.



Pie Chart The process is nothing new, so I only show the style part:

```
option = {
    tooltip: {
        trigger: 'item',
        formatter: "{a} <br/>{b}: {c} ({d}%)"
    },
    legend: {
        orient: 'vertical',
        x: 'left',
        data:['ECCV', 'NIPS', 'SIGKDD', 'WWW', 'SIGIR', 'CVPR', 'ICCV', 'NAACL', 'ICML', 'AAAI', 'ACL', 'EMNLP', 'IJCAI']
    },
    series: [
        {

            <--! Other Options -->

            data:[
                {value:75329, name:'A+', selected:true},
                {value:22886, name:'A'}
            ]
        },
        {

            <--! Other Options -->

            data:[
                {value:8906, name:'NIPS'},
                {value:6242, name:'SIGKDD'},
                {value:8893, name:'WWW'},
                {value:5050, name:'SIGIR'},
                {value:6893,name:'ICCV'},
                {value:6516,name:'ICML'},
                {value:12131,name:'AAAI'},
                {value:10623,name:'ACL'},
                {value:10075,name:'IJCAI'},
                {value:4579, name:'ECCV'},
                {value:11791,name:'CVPR'},
                {value:3974,name:'NAACL'},
                {value:2542,name:'EMNLP'},
            ]
```

```
41                  }
42              ]
43      };;
```



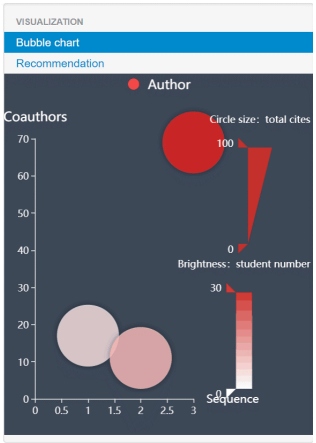
Bubble Chart For the bubble chart, considering which characteristics of the author we should use is of great importance. At last, we choose author sequence, author name, cites, author ID, coauthors, teachers and students these seven characteristics. Because cites and student number is more effective, we use them to decide brightness and bubble size.

```
1       var schema = [
2                   {name: 'authorsquence', index: 0,
                        text: 'Sequence'},
3                   {name: 'authorname', index: 1, text:
                         'Coauthors'},
4                   {name: 'cites', index: 2, text: '
                        Total cites'},
5                   {name: 'authorid', index: 3, text: '
                        ID'},
6                   {name: 'coauthors', index: 4, text:
                        'Name'},
7                   {name: 'teachers', index: 5, text: '
                        Teachers'},
8                   {name: 'students', index: 6, text: '
                        Students'} ];
9
10      <--! Other Options -->
```

The following code can add a demonstration tool to help user understand why the color and circle size are different:

```
    visualMap: [
        <--! Some Options -->
    ],
    series: [
        {
        name: 'Author',
        type: 'scatter',
        itemStyle: itemStyle,
        data: data
        }
    ]
```

After receiving data from back-end, bubble chart about a certain author has finished!



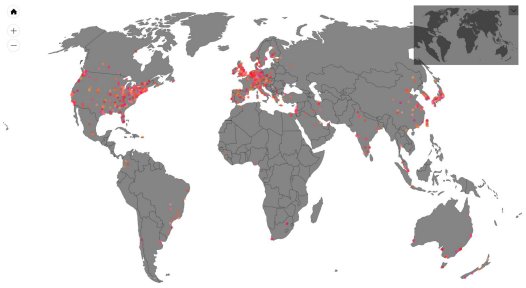
Map Map is more like extension of our website with more abundant data and more rich functions. So, I look for Acemap and find the world map from it. After extracting data of each point from our database, I use python to output these data – the number is just too large to do it by hands!

Then we can draw a map with the points we extract. What we have to do next is to set a world map as background and determine the location of these points. It is also necessary to ensure the point size appropriate.

```
        var minBulletSize = 6;
        var maxBulletSize = 11;
        var min = Infinity;
        var max = -Infinity;
```

```
5
6           AmCharts.theme = AmCharts.themes.black;
7
8             // get min and max values
9             for (var i = 0; i < mapData.length; i++) {
10              var value = mapData[i].value;
11              if (value < min) {
12                min = value;
13              }
14              if (value > max) {
15                max = value;
16              }
17            }
18
19            AmCharts.ready(function() {
20              map = new AmCharts.AmMap();
21              map.projection = "winkel3";
22
23              map.areasSettings = {
24                unlistedAreasColor: "#FFFFFF",
25                unlistedAreasAlpha: 0.1
26              };
27              map.imagesSettings = {
28                balloonText: "<span style='font-size
                      :14px;'><b>[[title]]</b></span>",
29                alpha: 0.6
30              }
31
32              var dataProvider = {
33                mapVar: AmCharts.maps.worldLow,
34                images: [],
35                getAreasFromMap:true
36              }
37
38                dataProvider.images.push({
39                  type: "circle",
40                  width: size,
41                  height: size,
42                  color: dataItem.color,
43                  longitude: latlong[id].longitude,
44                  latitude: latlong[id].latitude,
45                  title: id+dataItem.name,
46                  value: value
```

```
47                  });
48                }
49
50                map.dataProvider = dataProvider;
51                map.areasSettings = {
52                  autoZoom: true,
53                  rollOverBrightness:10,
54                  selectedBrightness:20
55                };
56
57                map.smallMap = new AmCharts.SmallMap();
58                map.write("mapdiv");
59              });
```



D3.js. D3.js is a useful JavaScript library for visualization. From the official website of D3, we can learn that D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as HTML, SVG, and CSS. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3's functional style allows code reuse through a diverse collection of official and community-developed modules.

To be honest, some graphs are inspired by the source code from D3.js, including force directed graph and tree graph. I will introduce them one by one.

Tree Graph First, we draw a tree without any node or link:

```
1       function tree() {
2           var _chart = {};
3           var _width = 400, _height = 800,
4                   _margins = {top: 30, left: 90, right
                        : 30, bottom: 30},
5                   _svg,
6                   _nodes,
7                   _i = 0,
8                   _tree,
9                   _diagonal,
10                  _bodyG;
11          _chart.render = function () {
12              if (!_svg) {
13                  _svg = d3.select("body").append("svg
                        ")
14                              .attr("height", _height)
15                              .attr("width", _width);
16              }
17              renderBody(_svg);
18          };
```

Here we only set the size of the SVG graphics. D3.layout.tree handles other things on its own and calculate the position of each node in turn. To use the tree, just call the nodes function.

```
1       function renderBody(svg) {
2       <--! Some Options -->
3           }
4           function render(source) {
5       <--! Some Options -->
6           }
7           function renderNodes(nodes, source) {
8       <--! Some Options -->
9       }
```

Now we use the generated nodes as data and use them to generate nodes in the tree. Use the index number as the ID of each node to ensure object persistence.

```
1       var nodeEnter = node.enter()
2             .append("svg:g")
3             .attr("class", "node")
4             .attr("transform", function (d)
5             {return "translate(" + source.y0  + "," +
                 source.x0 + ")";
6                     })
7             .on("click", function (d) {
8             toggle(d);
9             render(d);
10            });
11
12            nodeEnter.append("svg:circle")
13            .attr("r", 1e-6)
14            .style("fill", function (d) {
15            return d._children ? "lightsteelblue" : "#
                 fff";
16            });
17
18      var nodeUpdate = node.transition()
19            .attr("transform", function (d) {
20            return "translate(" + d.y + "," + d.x + ")
                 ";
21            });
22
23            nodeUpdate.select("circle")
24            .attr("r", 4.5)
25            .style("fill", function (d) {
26            return d._children ? "lightsteelblue" : "#
                 fff";
27            });
28
29      var nodeExit = node.exit().transition()
30            .attr("transform", function (d) {
31            return "translate(" + source.y
32            + "," + source.x + ")";
33            })
34            .remove();
35
36            nodeExit.select("circle")
37            .attr("r", 1e-6);
38            renderLabels(nodeEnter, nodeUpdate,
                 nodeExit);
```

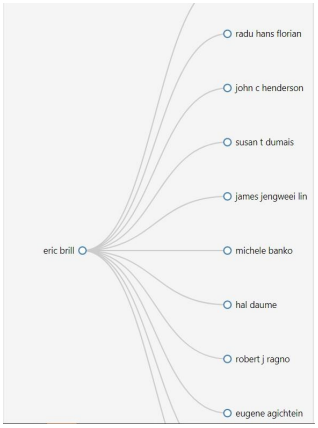
```
39              nodes.forEach(function (d) {
40              d.x0 = d.x;
41              d.y0 = d.y;
42              });
43              }
44
45      function renderLabels(nodeEnter, nodeUpdate,
            nodeExit) {
46      <--! Some Options -->
47          }
48
49      function renderLinks(nodes, source) {
50      <--! Some Options -->
51                      });
```

In the enter section, the svg:path element is created, which describes the connection between the source node and the destination node. The attribute d is also generated using the previously defined diagonal generator. During the process of generation, we will temporarily set the length of the connection line is set to d3.svg.diagonal, which is to put the source node and the target node in the same location, after which we adjust its length through animation effects.

```
1       link.enter().insert("svg:path", "g")
2                   .attr("class", "link")
3                   .attr("d", function (d) {
4       var o = {x: source.x0, y: source.y0};
5                           return _diagonal({source: o,
                                target: o});
6                   });
7               link.transition()
8                       .attr("d", _diagonal);
9               link.exit().transition()
10                      .attr("d", function (d) {
11                          var o = {x: source.x, y:
                               source.y};
12                          return _diagonal({source: o,
                                target: o});
13                      })
14                      .remove();
15          }
16          function toggle(d) {
```

```
17            <--! Some Options -->
18          }
19          function toggleAll(d) {
20          <--! Some Options -->
21          }
22          _chart.width = function (w) {
23              if (!arguments.length) return _width;
24              _width = w;
25              return _chart;
26          };
27          _chart.height = function (h) {
28          <--! Some Options -->
29          };
30          _chart.margins = function (m) {
31          <--! Some Options -->
32          };
33          _chart.nodes = function (n) {
34          <--! Some Options -->
35          };
36          return _chart;
37      }
38      var chart = tree();
```

Then we need to receive data to draw a complete tree, the required format is json with "'name' => and 'children' =>". To make the relationship clearer, I split it into student tree and teacher tree, both of which follow this pattern.



Force-Directed Graph At first, we set the style of link and node, then waiting for data to transfer. The process and principle is similar to tree

graph, so I will show some unique key code below.

```
1       var simulation = d3.forceSimulation()
2           .force("link", d3.forceLink().id(function(d)
                { return d.id; }))
3           .force("charge", d3.forceManyBody())
4           .force("center", d3.forceCenter(width / 2,
               height / 2));
5
6       var link = svg.append("g")
7           .attr("class", "links")
8           .selectAll("line")
9           .data(graph.links)
10          .enter().append("line")
11          .attr("stroke-width", function(d) { return
               Math.sqrt(d.value); });
12
13      var node = svg.append("g")
14          .attr("class", "nodes")
15          .selectAll("circle")
16          .data(graph.nodes)
17          .enter().append("circle")
18          .attr("r", 5)
19          .attr("fill", function(d) { return color(d.
               group); })
20          .call(d3.drag()
21          .on("start", dragstarted)
22          .on("drag", dragged)
23          .on("end", dragended));
```

2.3. Difficulties & Improvement

2.3.1. Automatic Completion

We did not modify the CSS file of the automatic completion box, thus its appearance is not very beautiful. The length of it is not fixed and its background color does not match the web background.

2.3.2. Type of Tables

The type of table is not rich enough.

We use several line charts to display our data, but in fact more choices are provided. For instance, radar chart and heat map are also optional forms when we display tags or conferences.

2.4. Function of Graphs

To make users learn more comprehensive information, function of each graph can be enriched. For example, we can show the exact number of each point and the maximum and minimum number in the line chart. And we can show the paper number and number of authors in the point of map as well. Additionally, the connection method of force graph can be improved, maybe we add three-dimensional effects or something.

3. Interaction between Back-end and Front-end

Since the framework is finished, the remaining part is to just feed data to our website.

3.1. Specific Implement

3.1.1. Homepage

The data needed in our homepage is mainly for the autoocomplete function.

Autocomplete. To realize the autocomplete function, we create the method "hint-xxx" in our Controller.

```
1      public function hint_author()
2          {
3              $q = strtolower($_GET["term"]);
4              $this->load->model('author_model');
5              $this->author_model->get_hint($q);
6      }
```

The method accepts the user's input, and load the method "get_hint" in "Author_model", which is defined as follows:

```
1       public function get_hint($q)
2       {
3           $params = [
4               'size' => 10,
5               'index' => 'authors',
6               'type' => 'test-type',
7               'body' => [
8                   'query' => [
9                       'prefix' => [
10                          'name.keyword' => $q
11                      ]
12                  ],
13                  'sort' => [
14                      'num_paper' => [
15                          'order' => 'desc'
16                      ]
17                  ]
18              ]
19          ];
20          $response = $this->client->search($params);
21             $data = $response['hits']['hits'];
22          foreach ($data as $item) {
23              $res[] = array(
24                  'id' => $item['_source']['id'],
25                  'label' => $item['_source']['name']
26              );
27          }
28          echo json_encode($res);
29      }
```

The method above performs the DSL to get the authors' data whose name begins with the parameter and sort them in order.

Other two "hint" methods use the similar DSL to collect the data for autocomplete function.

3.1.2. Result Page

Author. When we click the form's submit button in our homepage, we turn to the "author" method.

```
1       public function author()
2       {
3               $scholarname = $_GET["scholarname"];
4           $this->load->model('author_model');
5           $num = $this->author_model->get_num (
                $scholarname);
6           $data['scholarname'] = $scholarname;
7           $data['num'] = $num;
8           $this->load->view('authorview', $data);
9       }
```

The parameter "scholarname" is passed to the method. Then, to load basic data such as the number of pages in our search result page, the method invokes "get_num" to get the number of search results. In the end, data including the scholar's name and the number are sent to the "authorview", which will display the results piece by piece.

In "authorview", we use AJAX to collect the authors' information from the back end. Every time when the event is triggered, the function "getdata" defined in jQuery will ask method "author_data" to provide the needed data.

```
1       function getdata(){--
2           <! Some codes -->
3           $.getJSON("author_data", {'scholarname':'<?
                php echo $scholarname; ?>', 'page':page},
4           function(data){--
5               <! Some codes -->
6           })
7       }
```

And the method "author_data" will ask the data from "Author_model" with the current page number and then receive the data in JSON form. The data will straightly sent to the area needed to be filled in the result page.

```
1       public function author_data()
2       {
3           $name = $_GET["scholarname"];
4           $page = $_GET["page"];
5           $start = ($page - 1) * 10;
6           $this->load->model('author_model');
7           $this->author_model->get_last_ten_entries (
```

```
            $name, $start);
    }
```

```
    public function get_last_ten_entries ($name,
        $start)
    {
        $params = [
            'size' => 10,
            'from' => $start,
            'index' => 'my-index',
            'type' => 'test-type',
            'body' => [
                'query'=> [
                    'match_phrase'=> [
                        "name" => $name
                    ]
                ],
                'sort'=> [
                    'num_paper'=> [
                        'order'=> 'desc'
                    ]
                ],
                'highlight' => [
                    'pre_tags' => ["<span class='
                        my_font'>"],
                    'post_tags' => ["</span>"],
                    'fields' => [
                        'name' => new \stdClass()
                    ]
                ]
            ]
        ];
    }
```

And it's worth mentioning that, as the above code shows, after we do the query and sort, we also use the Elasticsearch's highlight function, automatically adding the pre-tags and post-tags for the search targets. So in the result pages, these content will be highlighted for better presentation.

Paper. We create "paperview" to present the results when the user searches for papers. And similar to "authorview", it's loaded by the method "paper"

in our Controller.

```
1       public function paper()
2       {
3           $papertitle = $_GET["papertitle"];
4           $this->load->model('paper_model');
5           $num = $this->paper_model->get_num(
                $papertitle);
6           $data['papertitle'] = $papertitle;
7           $data['num'] = $num;
8           $this->load->view('paperview', $data);
9       }
```

But this time we add a tendency chart of the numbers of papers in last 10 years in the paper's result pages. So there are extra data needed to be collected.

```
1       <iframe src="paper_graph?papertitle=<?php echo
            $papertitle; ?>" frameborder='0' width='400'
            scrolling='No' height='400' leftmargin='0'
            topmargin='0' id='paper_graph'></iframe>
```

The HTML "iframe" tag links to "paper_graph" in the Controller. This method will load "paper_year_graph_view", which will eventually be shown in the result pages.

```
1       public function paper_graph_data()
2       {
3           $papertitle = $_GET["papertitle"];
4           $this->load->model('paper_model');
5           $this->paper_model->get_year_data(
                $papertitle);
6       }
```

Conference. Since the information needed in our conference's result pages is little, our back end structure is designed in a simpler way. That is, we don't need to use AJAX but instead directly transmit all the data when loading "conferenceview".

```
1       public function conference()
2       {
3           $conference = $_GET["conference"];
4           $list = array('ECCV', 'NIPS', 'SIGKDD', 'WWW
                ','SIGIR','CVPR','ICCV','NAACL','ICML','
               AAAI','ACL','EMNLP','IJCAI');
5           if (!in_array($conference, $list))
6           {
7               $this->load->view('errors/error_404');
8           }
9           else
10          {
11              $this->load->model('conference_model');
12              $data = $this->conference_model>
                    get_result($conference);
13              $this->load->view('conferenceview',
                    $data);
14          }
15      }
```

3.1.3. Author Page

The basic structure including the author's specific information and teacher/student tree graphs is similar to the result pages, so we will not repeat it now. What we want to mention is the Force-Directed graph.

In "author_ind_view", we set an "iframe" tag which links to "force_graph" method:

```
1       public function force_graph()
2       {
3           $authorid = $_GET["id"];
4           $data['authorid'] = $authorid;
5           $this->load->view('force_graph_view', $data)
                ;
6       }
```

And in "force_graph_view", we request the data from "force_graph_data":

```
1       d3.json("author_graph_data?authorid=<?php echo
            $authorid; ?>", function(error, graph) {
2       })
```

And the method “author_graph_data” invokes “get_graph_data” to collect the Force-Directed graph data.

```
1       public function author_graph_data()
2       {
3           $authorid = $_GET['authorid'];
4           $this->load->model('author_model');
5           $this->author_model->get_graph_data(
                $authorid);
6       }
```

“get_graph_data” is quite complicated. Unlike the one-direction relationship in student/teacher trees, in the Force-Directed graph, we need to determine relationships between every pair of coauthors. So it’s inevitable to execute DSL more than once.

We first find all the cooperators of the author, and create the node of the author.

```
1       $params = [
2           'index' => 'my-index',
3           'type' => 'test-type',
4           'body' => [
5               'query'=> [
6                   'term'=> [
7                       "id.keyword" => $authorid
8                   ]
9               ]
10          ]
11      ];
12
13      $response = $this->client->search($params);
14      $data = $response['hits']['hits'];
15      foreach ($data as $item) {
16          $res = $item['_source'];
17      }
18
19      $students = $res['student'];
20      $teachers = $res['teacher'];
21      $coauthors = $res['coauthor'];
22      $nodes[] = array(
23          'id' => $res['id'],
24          'name' => $res['name'],
```

```
25              'group' => 1
26          );
```

And then, for each kind of cooperators (student, teacher, coauthor), we use a loop to create nodes and links between the cooperators and the author. We only take student type as an example.

```
1       foreach ($students as $student) {
2           $params2 = [
3               'index' => 'my-index',
4               'type' => 'test-type',
5                   'body' => [
6                       'query'=> [
7                           'term'=> [
8                               "id.keyword" => $student
9                           ]
10                      ]
11                  ]
12          ];
13
14          $response2 = $this->client->search($params2)
                ;
15          $data2 = $response2['hits']['hits'];
16          foreach ($data2 as $item2) {
17              $res2 = $item2['_source'];
18          }
19
20          $nodes[] = array(
21              'id' => $res2['id'],
22              'name' => $res2['name'],
23              'group' => 2
24          );
25
26          $links[] = array(
27              'source' => $res['id'],
28              'target' => $res2['id'],
29              'value' => 2
30          );
31      }
```

Now, we've got all the nodes and the links between cooperators and the

author. It's time to determine the relationship among these cooperators and add links. We use a nested loop to realize it.

```
1       $len = count($nodes);
2       for ($i = 1; $i < $len; $i++){
3           $params5 = [
4               'index' => 'my-index',
5               'type' => 'test-type',
6               'body' => [
7                   'query'=> [
8                       'term'=> [
9                           "id.keyword" => $nodes[$i]['
                                id']
10                      ]
11                  ]
12              ]
13          ];
14
15          $response5 = $this->client->search($params5)
                ;
16          $data5 = $response5['hits']['hits'];
17          foreach ($data5 as $item5) {
18              $res5 = $item5['_source'];
19          }
20
21          for ($j = $i + 1; $j < $len; $j++){
22              if (in_array($nodes[$j]['id'], $res5['
                    student'])
23              || in_array($nodes[$j]['id'], $res5['
                    teacher'])
24              || in_array($nodes[$j]['id'], $res5['
                    coauthor'])){
25                  $links[] = array(
26                      'source' => $nodes[$i]['id'],
27                      'target' => $nodes[$j]['id'],
28                      'value' => 2
29                  );
30              }
31          }
32      }
```

And finally, we integrate the data into one array, and transform it to the

JSON form.

```
1       $result = array('nodes' => $nodes, 'links' =>
            $links);
2
3       echo json_encode($result);
```

3.1.4. Paper Page

Apart from the basic information of the paper, in the paper page, we provide the user with our recommendation of other good papers and visualize the authors' information with the bubble chart. Another function fulfilled is the tagging system, which is purposely designed to optimize the user experience.

Recommendation & Tagging system. Before we contrive the structure of the recommendation module, we need to first realize the users' needs. When they are searching for a certain paper by its whole title or a part of it, they often have an interest to learn more about the related area. For instance, if a user is attracted by a paper in the author page, he may enter the paper page for further information. It's worth mentioning that in such a case, he only focuses on the content or just the topic of the paper. In other words, he finds a good "word", and wants to know more about it.

But what exactly is a good "word"? How do we help him discovery more? "Tagging system" is our solution. (Due to the limited time, we only fulfill part of the system. There're only one type of tags, which is of computer science. We will discuss it later in the last chapter.)

The tagging system is based on the data of hot keywords in computer science, which is provided by Acemap. And for these words, we extract them from the papers' title, and set them as a field of the paper-type documents. Then, for every paper, we can tag them with a certain keyword, such as "machine learning".

We further group the papers by their tags. In the paper page, if the user clicks the tag, the page will turn to the tag page, where the most popular papers in this area as well as the development tendency will be shown.

Let us return our attention to the recommendation. With the help of tagging system, we can easily recommend those papers with the same tags. Besides, we also recommend the papers of the same authors, which we consider also a good aspect of recommendation.

Since the structure of DSL is mainly the same as the former ones, we will not show it here.

Bubble Chart. We this time again use the tag "iframe" to control the bubble chart. It will accept the data transmitted from "paper_bubble_chart" in Controller.

```
1       public function paper_bubble_chart()
2       {
3           $paperid = $_GET["id"];
4           $this->load->model('paper_model');
5           $paperdata = $this->paper_model->
                get_ind_data($paperid);
6           $authors = $paperdata['authorinfo'];
7           $this->load->model('author_model');
8           foreach ($authors as $author) {
9               $data['authorinfo'][$author['
                    authorsequence']] = $this->
                    author_model->get_ind_data($author['
                    authorid']);
10          };
11          $this->load->view('bubble_chart_view', $data
                );
12      }
```

Instead of defining another method, we directly use the method "get-int-data" in "Author_model" to collect the author data.

3.2. Difficulties & Improvement

3.2.1. Tagging System

Due to the lack of data, we only set one type of tag. We may extract our own hot keywords from other areas, which may need some knowledge of data analysis.

3.2.2. Visualization with Kibana

As we mention above, Kibana is a powerful visualization platform. But we only use it as a tool to interact with Elasticsearch.

Since our data are static, it may be unnecessary to use Kibana to analyze. Nevertheless, we will try to use it to deal with the cases when the data feature is unclear. Besides, we can use the combination of Elasticsesarch, Logstash and Kibana to build the whole centralized blogging platform to support our website.

4. Acknowledgement

It takes enormous efforts of the entire team to develop the website. We truly discussed and coded together for days. Every member is irreplaceable.

Yixiong and Jiasong do a great job in beautifying our web page, making it clean but informative. They pointe to positive directions in which the others could easily fulfill the details.

Ruoyu is in charge of Visualization. He figures out the best way to enrich the information based on the need of users, and provides a lot of interesting functions.

Zhihui is responsible for the back-end integration and data transmission. With his code as the starting point, he further integrates all the components into MVC framework. He also uses Elasticsearch to boost the web.

And we're thankful for the data provided by Acemap. Without these data, a lot of functions may not be realized.