Final Project

Group 2

Shen Tianxiao & Li Ziyi & Jing Hongyi & Pang Lianyu

June 23, 2018

Contents

1 Introduction

1.1 Problem Analysis

1.1.1 Project and Mission

Main Project: In the final project, the four of us integrated the functions of the previous four experiments,and formed a relatively complete academic searching website, and optimized the relevant pages.

Final Project 1:Previous search pages could only search by scholar's name. In this search results page, we added ways of searching for paper title and conference.Users can select different single boxes on the homepage to realize different search types.

Final Project 2:After adding the search types of paper title and conference, the paper and conference pages are added to the website to display relevant information.

Final Project 3:When presenting information on each page, if more information items are displayed (such as search results, papers), then it should display 10 pieces per page, and add page turning function.

Final Project 4:Article recommendations (such as through co-authors, co-references, etc.),which can be made in the paper.The related papers can be recommended on the page, and the corresponding papers can also be submitted on the scholar's page.

Final Project 5:The tree structure of teacher relationship can be displayed, and the tree can be opened layer by layer.Also we consider the case where a scholar may have multiple mentors.

Final Project 6:At the end of the site,in the authors page we added scholar relation network, and according to the relationship between the scholars they are classified for the "student", "teacher", and "others" three types and use different colors to distinguish them.

1.1.2 Preparations

Tools to prepare:In this experiment, we mainly used two compilers, Phpstorm and Notepad++ +, and used Dreamweaver for real-time displayment and adjustment of website effects.We still use Mysql as the data repository, and use CSS, Javascript, and HTML to orchestrate and tweak web pages.

Beautification and user interface:The homepage mainly uses HTML for the layout of the webpage, and we used a video as the background, which is dynamic and beautiful.In the scholar information page and the paper page, the yellow color is used as the background and the blue text is used to make the whole page more concise and intuitive.

Learn to use D3.js:D3.js is a JavaScript library for manipulating documents

based on data. D3 helps you bring data to life using HTML, SVG, and CSS. D3s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.
Modifying documents using the W3C DOM API is tedious: the method names are verbose, and the imperative approach requires manual iteration and bookkeeping of temporary state. For example, to change the text color of paragraph elements:

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
var paragraph = paragraphs.item(i);
paragraph.style.setProperty("color", "white", null);
}
```

D3 employs a declarative approach, operating on arbitrary sets of nodes called selections. For example, you can rewrite the above loop as:

```
3.selectAll("p").style("color", "white");
```

Yet, you can still manipulate individual nodes as needed:

```
d3.select("body").style("background-color", "black");
```

Updating nodes are the default selectionthe result of the data operator. Thus, if you forget about the enter and exit selections, you will automatically select only the elements for which there exists corresponding data. A common pattern is to break the initial selection into three parts: the updating nodes to modify, the entering nodes to add, and the exiting nodes to remove.

```
// Update...
var p = d3.select("body")
.selectAll("p")
.data([4, 8, 15, 16, 23, 42])
.text(function(d) { return d; });

// Enter...
p.enter().append("p")
.text(function(d) { return d; });

// Exit...
p.exit().remove();
```

1.2 Result Display

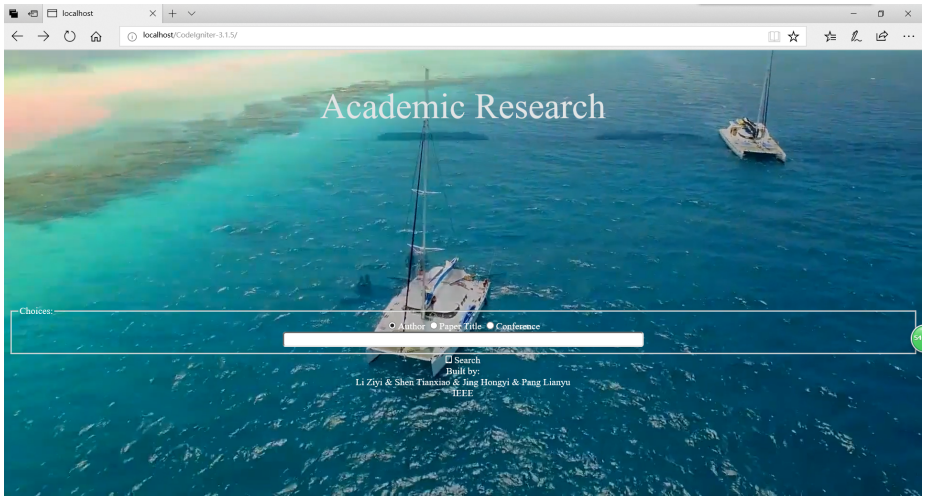


Figure 1: home.php

Figure 2: paper.search.php

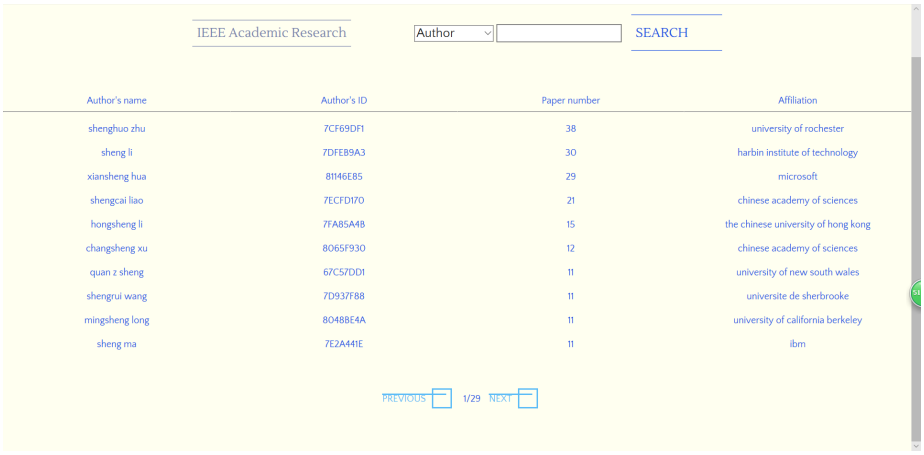


Figure 3: paper.php

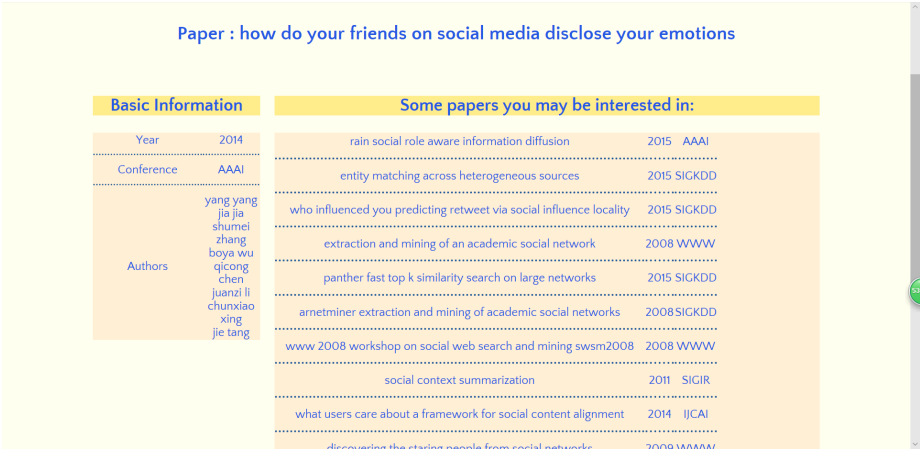
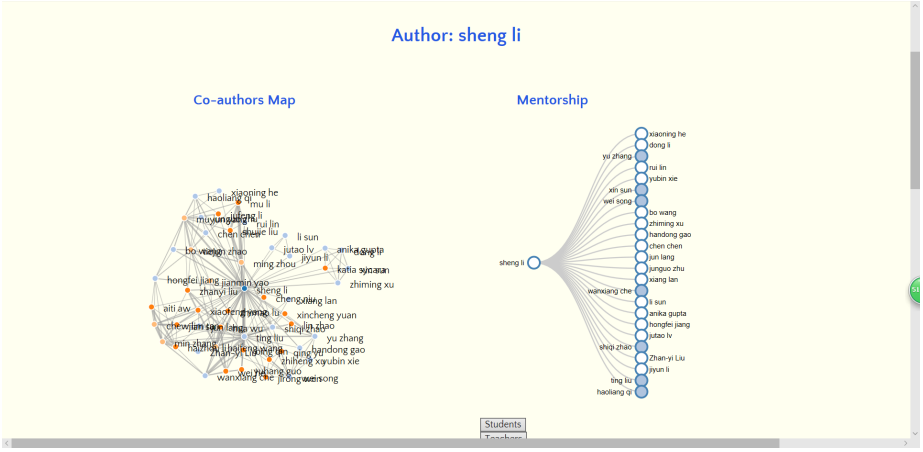


Figure 4: author.search.php



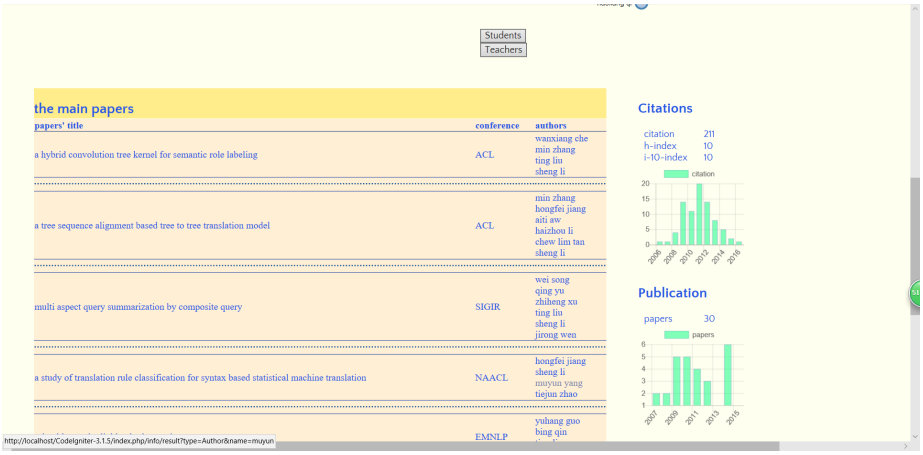


Figure 5: author.php

Figure 6: conference.search.php

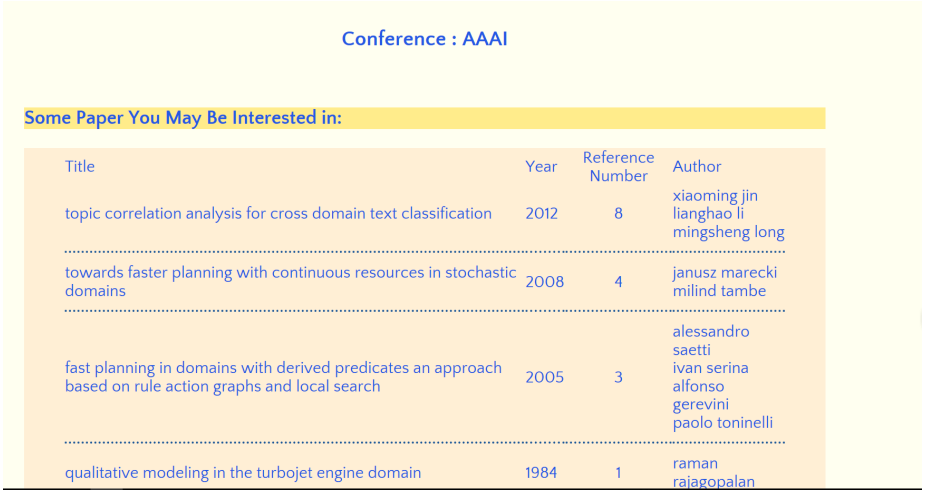


Figure 7: conference.php

2 Specific Work

2.1 Functionalities

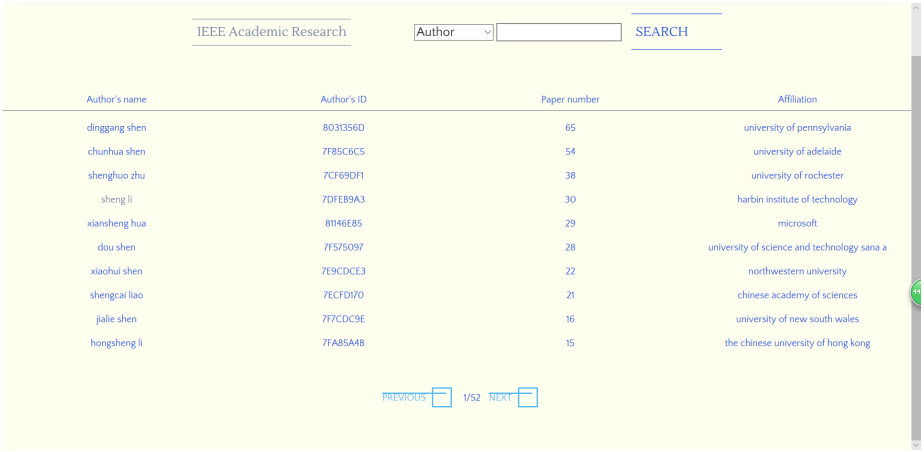
2.1.1 Different Types of Searches

Type 1:Author Search
Select the Author radio button on the homepage, enter the scholar's name and search. The scholar's paper information is displayed on the page. Ten papers are displayed on each page and sorted according to the number of references. The acceptance of the author's page data and the generating table code are:

```
<?php
$referPaper=$data['referPaper'];
$authors=$data['authors'];
$papers=$data['papers'];
$name=$data['name'];
echo "<tr>";
for($i=0;$i<4;$i++)echo"<td><hr> </td>";
echo "</tr>";
echo "<tr><td width=200 height=100 align='center'><a href=result?
    type=Author&name=$name[authorsname] methods='get'>$name[
    authorsname]</a></td>";
$i=0;
foreach ($referPaper as $value)
{
if($i!=0)echo"<td></td>";
$i++;
foreach ($papers[$value] as $paper) {
$j = 0;
foreach ($paper as $item) {
if ($j = 0)
echo "<td width=500 height=100>$item</td>";
else echo "<td width=400 height=100 align='center' '>$item</td>";
$j++;
}
echo "<td>";
foreach ($authors[$value] as $item)
foreach ($item as $author)
echo "<a href=result?type=Author&name=$author methods='get'>$author
    </a><br/>";
echo "</tr>";
echo "<tr>";
for ($i = 0; $i < 3; $i++) echo '<td><hr style=" height:2px;border:
    none;border-top:2px dotted #185598;" /></td>';
echo "</tr>";
}
}
?>
</table>
```

This is the result displayment of the author search:

Type 2:Paper Title Search
On the main page to select Paper Title radio button, the input to search after the Paper topics, in the page displays information Paper, each page shows ten papers and according to the reference number to sort, and we had a Paper in web page recommendation.The acceptance of the data of the paper title page and the generating form code are:



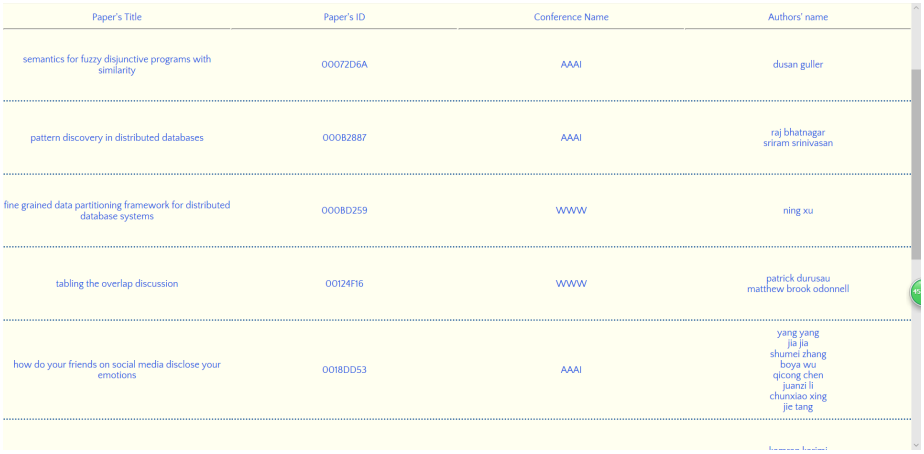

```
<?php
echo'<table width=1600>';
echo"<tr><td width=400>Title</td><td width=1200>$papers[Title]</td>
    </tr>";
echo"<tr><td width=400>Paper Published Year</td><td width=1200>
    $papers[PaperPublishYear]</td> </tr>";
echo"<tr><td width=400>Conference's Name</td><td width=1200>$papers
    [ConferencesName]</td> </tr>";
echo"<tr><td width=400>Authors' Name</td><td width=1200>";
foreach ($authors as $author)
echo $author[0]."<br/>";
echo "</td> </tr><tr>";
for ($i = 0; $i < 2; $i++) echo "<td><hr/></td>";
echo "<tr/>";
echo"</table>";
echo'<table width=1600 id="tb">';
foreach ($referPapers as $referPaper){
if($i==2) {
echo "<tr><td width=400>Refer Paper</td>";
$i=3;
echo "<td width=1200>$referPaper[0]</td></tr>";
}
else {
echo"<tr><td width=400></td>";
echo "<td width=1200>$referPaper[0]</td></tr>";
}
}
echo"</table>";
?>
```

This is the result of the paper title search:

Type 3:Conference Search

In the main page to select the Conference radio button, the input to search after the name, the page displays information about the papers published in the institution, and each page shows ten papers in accordance with the reference number to sort, and in web page recommendation we made in the paper.The acceptance of the data of the paper title page and the generating form code are:

```
<?php
echo "<tr>";
for($i=0;$i<4;$i++)echo"<td><hr> </td>";
```



Paper's Title	Paper's ID	Conference Name	Authors' name
semantics for fuzzy disjunctive programs with similarity	00072D6A	AAAI	dusan guller
pattern discovery in distributed databases	000B2887	AAAI	raj bhatnagar sriram srinivasan
fine grained data partitioning framework for distributed database systems	000BD259	WWW	ning xu
tabling the overlap discussion	00124F16	WWW	patrick durusau matthew brook odonnell
how do your friends on social media disclose your emotions	0018DD53	AAAI	yang yang jia jia shumei zhang boya wu qicong chen juanzi li chunxiao xing jie tang

```
echo "</tr>";
echo "<tr><td width=200 height=100 align='center'><a href=result?
    type=Author&name=$name[authorsname] methods='get'>$name[
    authorsname]</a></td>";
$i=0;
foreach ($referPaper as $value)
{
if($i!=0)echo"<td></td>";
$i++;
foreach ($papers[$value] as $paper)
$j = 0;
foreach ($paper as $item) {
if ($j = 0)
echo "<td width=500 height=100>$item</td>";
else echo "<td width=400 height=100 align='center' '>$item</td>";
$j++;
}
echo "<td>";
foreach ($authors[$value] as $item)
foreach ($item as $author)
echo "<a href=result?type=Author&name=$author methods='get'>$author
    </a><br/>";
echo "</tr>";
echo "<tr>";
for ($i = 0; $i < 3; $i++) echo '<td><hr style=" height:2px;border:
    none;border-top:2px dotted #185598;" /></td>';
echo "</tr>";
}
}
?>
```

This is the result of the conference search:

2.1.2 Paper Recommendation

processing In this section, we need to recommend some papers on the paper.php. These papers must be related to the main paper we searched. To achieve this, we decide to create a array which I used to store papers' ID and the corresponding number. Then we get the main paper's authors and searched these authors' papers respectively, putting them into the array. After that, we

make a loop to calculate the number of each paper. That is, if the paper has one same authors with the main paper, its number will be plus one. In the last, we reorder the array by the number DESC, and display their title, publishyear and conference for recommendation.

```
$paper = array();
  $paper[] = $paperid;
  $paper[$paperid] = 0;
  $temp=$paper;
  foreach($authorss as $aid)
  {
    $paper=$temp;
    $mysql2=$connect->query("SELECT PaperID FROM
    paper_author_affiliation WHERE AuthorID='$aid'");
    $row_z = $mysql2->fetch_all();
    foreach($row_z as $row)
    {
      $v=0;
      $k = $row[0];
      foreach($paper as $pid=>$h)
      {
        if($pid==$k)
        {
          $temp[$pid]++;
          $v=1;
        }
      }
      if($v==0)
      {
        $temp[$k]=1;
      }
    }
  }
arsort($temp);
```

2.2 Interface Design

2.2.1 homepage

On the academic search homepage, we used a short beach video as the background to make the entire page dynamic.With the short video as the background, the other elements are designed with simplicity in our mind.The implementation of the search function is placed in the lower center of the page. For the searchtype,we used three radio buttons to facilitate the user's choice.Below the search box are some basic information.

2.2.2 author,paper,conference

The scholar page, the paper page and the organization information page all adopt the relatively simple layout and displayment, and we use the relatively prominent light yellow static background. In the coming we added the search box to the top of the page (to save space, we have adopted the drop-down menu), which enables users to query in mid-flight and other scholars or paper information and quit the search service.

2.2.3 co-auhors

At the bottom of the scholar information page, we added the scholar cooperation diagram, which adopted the central typesetting to minimize itself, making the whole picture more compact and organized. Scholars who have the cooperation relationship have black lines linked together, and we can move the mouse to the scholars icon to show and search the relationship of scholars (teachers or students), and they have the different colors to be distinguished out. The whole picture is integrated and can be moved by dragging one of the scholars with the mouse.

2.2.4 mentorship

At the bottom of the scholar information page, there is also a teacher relationship tree, which is distributed as a horizontal tree graph. Each point represents a scholar.By clicking on different scholars, we can display the next level of the relationship tree, and so on, to achieve dynamic changes.

2.3 Data Visualization

2.3.1 Co-author Map

Graph To draw this graph, we need to use d3.js, which is intended to visualizing data. And then create a Scalable Vector Graphics.

```
<script type="text/javascript" src="http://localhost/CodeIgniter
    -3.1.5/jquery-3.3.1.js"></script>
<svg width="1500" height="600" id="container"></svg>
<script src="https://d3js.org/d3.v4.min.js"></script>
```

This code imports jquery and d3.v4.js, and set a svg so that we can draw on it. Then we use d3.js to describe the graph.

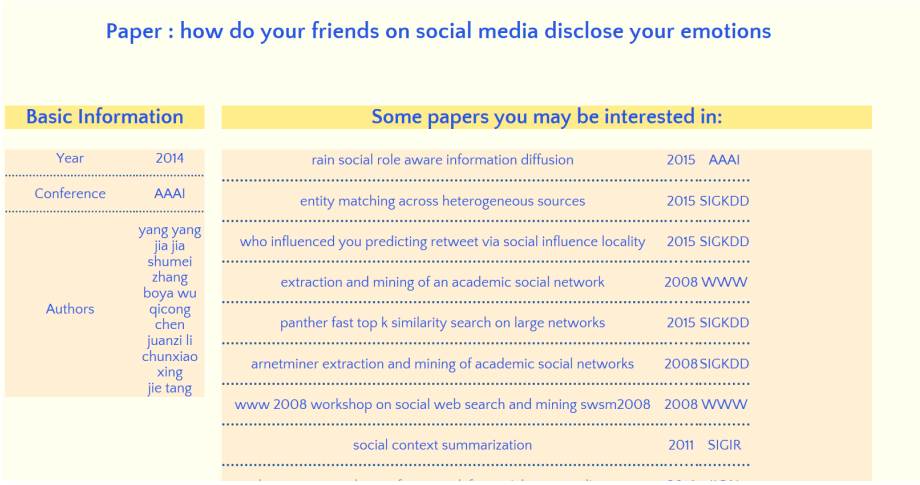


Figure 8: The papers we recommend

```
let svg = d3.select("svg"),
width = +svg.attr("width"),
height = +svg.attr("height");

let color = d3.scaleOrdinal(d3.schemeCategory20);

let simulation = d3.forceSimulation()
.force("link", d3.forceLink().id(function(d) { return d.id; }))
.force("charge", d3.forceManyBody().strength(function () {
return -100
}))
.force("center", d3.forceCenter(width / 2, height / 2));
```

This selects svg from the page, and sets the color, and the parameters that are necessary in drawing a force directed graph.

```
nodes=<?php echo json_encode($json['nodes']);?>;
links=<?php echo json_encode($json['links']);?>;

let link = svg.append("g")
.attr("class", "links")
.selectAll("line")
.data(links)
.enter().append("line")
.attr("stroke-width", function(d) { return Math.sqrt(d.value); });

let node = svg.append("g")
.attr("class", "nodes")
.selectAll("circle")
.data(nodes)
.enter().append("circle")
.attr("r", 5)
.attr("fill", function(d) { return color(d.group); })
.call(d3.drag()
.on("start", dragstarted)
.on("drag", dragged)
.on("end", dragended));

```

This code gets the data of links and nodes, and tells the computer how to draw the nodes and the links. "Enter()" can get the redundant data and put it into the right place. "Dragstarted","dragged" and "dragended" are three functions that will be defined later.

```
let texts=svg.selectAll("text")
.data(nodes)
.enter()
.append("text")
.style("fill","black")
.attr("dx",20)
.attr("dy",8)
.text(function (d) {
return d.name;
});

node.append("title")
.text(function(d) { return d.type; });

```

This connect the nodes with its name and type, so that the user can see them.

```
simulation
.nodes(nodes)
.on("tick", ticked);

simulation.force("link")
.links(links);

```

This code draws the graph.

```
function ticked() {
link
.attr("x1", function(d) { return d.source.x; })
.attr("y1", function(d) { return d.source.y; })
.attr("x2", function(d) { return d.target.x; })
.attr("y2", function(d) { return d.target.y; });

node
.attr("cx", function(d) { return d.x; })
.attr("cy", function(d) { return d.y; });

texts.attr("x",function (d) {
return d.x;
})
.attr("y",function (d) {
return d.y;
});
}

function dragstarted(d) {
if (!d3.event.active) simulation.alphaTarget(0.3).restart();
d.fx = d.x;
d.fy = d.y;
}

function dragged(d) {
d.fx = d3.event.x;
d.fy = d3.event.y;
}

function dragended(d) {
```

```
if (!d3.event.active) simulation.alphaTarget(0);
d.fx = null;
d.fy = null;
}

```

Here we define 4 functions, which can draw the graph when the mouse drag the nodes.

Getting data This subsection tells how to get the data we need to draw a graph.

```
$connect=new mysqli('localhost','root','','main_db');
if($connect->connect_error)
die("fail to connect.".$connect->connect_error);

```

To begin, we connect to the database.

```

$preTeachers=$connect->query("select teacherid
from teacher_and_student
where StudentID='$authorid'");
$preTeachers=$preTeachers->fetch_all();
$teachers=array();
foreach ($preTeachers as $preTeacher) {
$teachers[] = $preTeacher[0];
}
```

The table "teacher_and_student" stores all the teacher-student relationship, and how to get it is another story, which we will talk about later. This code selects the author's teachers.

```
$preStudents=$connect->query("select studentid
from teacher_and_student
where TeacherID='$authorid'");
$preStudents=$preStudents->fetch_all();
$students=array();
foreach ($preStudents as $preStudent){
$students[]=$preStudent[0];
}
```

This code selects all the author's students.

```

$paperid=$connect->query("select paperid
from paper_author_affiliation
where authorid = '$authorid'");
$paperid=$paperid->fetch_all();


foreach ($paperid as $value){
$authors[$value[0]]=$connect->query("select authorid
from paper_author_affiliation
where paperid ='$value[0]'");
}


$writers=array();
foreach ($authors as $author){
foreach ($author as $x=>$id) {
$writers[]=$id['authorid'];
}
```

```
}
$writers=array_unique($writers);

```

This code can get all the authors that have cooperated ever and put them into an array. The notes above have told how to get it.

```

$result=array();
foreach ($writers as $writer)
{
$template=array();
$authors0=array();
foreach ($writers as $w)
$template[$w]=0;

$paperid0=$connect->query("select paperid
from paper_author_affiliation
where authorid = '$writer'");
$paperid0=$paperid0->fetch_all();

foreach ($paperid0 as $value){
$authors0[]=$connect->query("select authorid
from paper_author_affiliation
where paperid = '$value[0]'");
}

foreach ($authors0 as $item){
foreach ($item as $x=>$id) {
foreach ($writers as $w){
if($id['authorid'] == $w){
$template[$w]++;
break;
}
}
}
}
$result[$writer]=$template;

```

This code calculate the number of papers that published by cooperation between this scientist and another one. "$template" is an array that stores the number in the form "authorID =¿ number".

```

$nodes=array();
$links=array();
foreach ($writers as $writer)
{
if($writer==$authorid){
$nodes[]=array("id"=>"$authorid","group"=>0,"name"=>$names[$writer
    ],"type"=>"author");
}
elseif(in_array($writer,$teachers)){
$nodes[]=array("id"=>"$writer","group"=>1,"name"=>$names[$writer],"
    type"=>"teacher");
}
elseif (in_array($writer,$students)){
$nodes[]=array("id"=>$writer,"group"=>2,"name"=>$names[$writer],"
    type"=>"student");
}
else{
```

```
$nodes[]=array("id"=>$writer,"group"=>3,"name"=>$names[$writer],"
    type"=>"others");
}
foreach ($result[$writer] as $id=>$value){
if ($value != 0) {
$links[] = array("source" => "$writer", "target" => "$id", "value"
    => $value);
}
}
}

for($i=0;$i<count($links);$i++){
for($j=0;$j<count($links);$j++){
if($links[$i]['target']==$links[$j]['source']and $links[$i]['source
    ']==$links[$j]['target'])
array_splice($links,$j,1);
}
}
return $json=array("nodes"=>$nodes,"links"=>$links);

```

This code sort the data we get and turns it into the form we need. In the nodes part, it has four parameters, and in the links part, it tells the computer how to link one node to another.

The table We use python to get the data and store it into database.

```
import pymysql
import numpy as np
import sklearn
import time

db = pymysql.connect(host='localhost', user='root', db='main_db',
    charset='utf8', port=3306,
cursorclass=pymysql.cursors.Cursor)
cursor = db.cursor()

```

We start with connecting the database by python.

```
def createTable():
cursor.execute("DROP TABLE IF EXISTS teacher_and_student")
create_table = '''create table teacher_and_student(
TeacherID char(8) not null ,
StudentID char(8) not NULL )DEFAULT CHARSET=utf8'''
cursor.execute(create_table)

createTable()

```

This code create the table.

```
papers=[]
teacher=[]
student=[]
clf = sklearn.externals.joblib.load('models/gnb.model')
with open(r'\data\papers.txt',encoding='utf8') as file:
while True:
content = file.readline()
if not content:
break
papers.append(content.split('\t')[0])

```

Here we get all the papers.

```
k=0
for paper in papers:
id1="""select authorid
from paper_author_affiliation
where PaperID='{}'""".format(paper)
cursor.execute(id1)
id1=cursor.fetchall()
id2=id1[:]
print(k)
k+=1
for i in range(len(id1)):
for j in range(i+1,len(id2)):
data = np.array([feature(id1[i][0], id2[j][0])]).reshape(1, -1)
result = clf.predict(data)
if(int(result) == 1):
teacher.append(id1[i][0])
student.append(id2[j][0])

```

Here we select all the authors that have ever cooperated with each other in a paper, and predict their relationship using the model we trained in the exercise 3.

```
for i in range(len(teacher)):
insert_data="""insert into teacher_and_student(TeacherID,StudentID)
     values ('{}','{}')""".format(teacher[i],student[i])
cursor.execute(insert_data)

db.commit()

```

And finally, we insert the data into the table and commit it.

2.3.2 Mentorship

processing Another work we did in data visualization is the mentorship diagram. We can easily come up with the idea that using a tree diagram to display the hierarchical teacher-student relationship. We used the tree diagram in d3.js. The nodes represent authors, and the links represent their relationship. Nodes on the right are the students of the node on the left. The tree is also interactive, i.e. it can be expanded layer by layer. If we click on the nodes, we can fold or unfold the diagram.
However, as we said in the presentation, one student may have multiple teachers, which means one child node may have multiple parents. And the structure of the tree only permits single parent. We can only know one's students and his/her students' students. How to deal with this problem is tricky, and due to our poor JavaScript experience we failed to solve it. And we are also confused about how to display the root author's students and teachers together. So finally we just provided two trees, one for students and the other for teachers.

```

    // Set the dimensions and margins of the diagram
  var margin = {top: 20, right: 90, bottom: 30, left: 90},
    width1 = 960 - margin.left - margin.right,
    height1 = 500 - margin.top - margin.bottom;

```

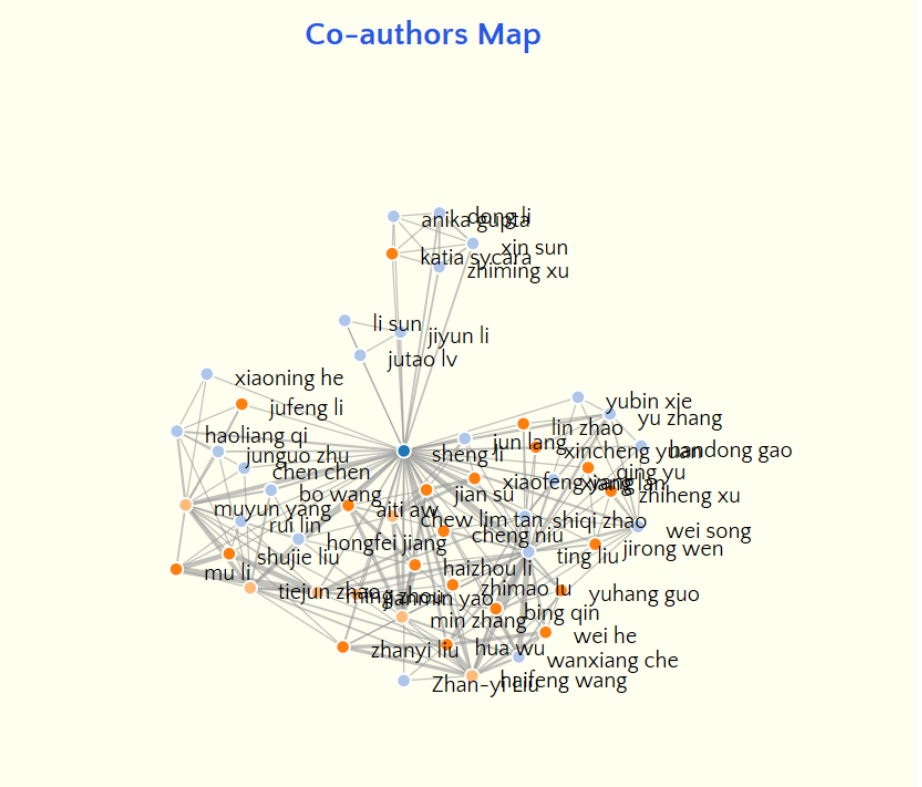


Figure 9: Co-author Map

```
// append the svg object to the body of the page
// appends a 'group' element to 'svg'
// moves the 'group' element to the top left margin
var svg1 = d3.select("#mentorship").append("svg")
  .attr("width", width1 + margin.right + margin.left)
  .attr("height", height1 + margin.top + margin.bottom)
  .append("g")
  .attr("transform", "translate("
        + margin.left + "," + margin.top + ")");

var i = 0,
  duration = 750,
  root;

// declares a tree layout and assigns the size
var treemap = d3.tree().size([height1, width1]);

var id="<?php echo $authorid;?>";

d3.json("http://localhost/CodeIgniter-3.1.5/mentorship.php?type=
  students&id="+id,function(error,treeData){
// Assigns parent, children, height, depth
  root = d3.hierarchy(treeData, function(d) { return d.children;
  });
  root.x0 = height / 2;
  root.y0 = 0;

```

```
  // Collapse after the second level
  root.children.forEach(collapse);

  update(root);
});
// Collapse the node and all it's children
function collapse(d) {
if(d.children) {
  d._children = d.children;
  d._children.forEach(collapse);
  d.children = null;
  }
}

function update(source) {

// Assigns the x and y position for the nodes
var treeData = treemap(root);

// Compute the new tree layout.
var nodes = treeData.descendants(),
    links = treeData.descendants().slice(1);

// Normalize for fixed-depth.
nodes.forEach(function(d){ d.y = d.depth * 180});

// ****************** Nodes section ***************************

// Update the nodes...
var node = svg1.selectAll('g.node')
    .data(nodes, function(d) {return d.id || (d.id = ++i); });

// Enter any new modes at the parent's previous position.
var nodeEnter = node.enter().append('g')
    .attr('class', 'node')
    .attr("transform", function(d) {
      return "translate(" + source.y0 + "," + source.x0 + ")";
  })
  .on('click', click);

// Add Circle for the nodes
nodeEnter.append('circle')
    .attr('class', 'node')
    .attr('r', 1e-6)
    .style("fill", function(d) {
        return d._children ? "lightsteelblue" : "#fff";
    });

// Add labels for the nodes
nodeEnter.append('text')
    .attr("dy", ".35em")
    .attr("x", function(d) {
        return d.children || d._children ? -13 : 13;
    })
    .attr("text-anchor", function(d) {
        return d.children || d._children ? "end" : "start";
    })
    .text(function(d) { return d.data.name; });

// UPDATE
var nodeUpdate = nodeEnter.merge(node);

```

```
94  // Transition to the proper position for the node
95  nodeUpdate.transition()
96    .duration(duration)
97    .attr("transform", function(d) {
98        return "translate(" + d.y + "," + d.x + ")";
99     });
100
101 // Update the node attributes and style
102 nodeUpdate.select('circle.node')
103   .attr('r', 10)
104   .style("fill", function(d) {
105       return d._children ? "lightsteelblue" : "#fff";
106   })
107   .attr('cursor', 'pointer');
108
109
110 // Remove any exiting nodes
111 var nodeExit = node.exit().transition()
112     .duration(duration)
113     .attr("transform", function(d) {
114         return "translate(" + source.y + "," + source.x + ")";
115     })
116     .remove();
117
118 // On exit reduce the node circles size to 0
119 nodeExit.select('circle')
120   .attr('r', 1e-6);
121
122 // On exit reduce the opacity of text labels
123 nodeExit.select('text')
124   .style('fill-opacity', 1e-6);
125
126 // ***************** links section ***************************
127
128 // Update the links...
129 var link = svg1.selectAll('path.link')
130     .data(links, function(d) { return d.id; });
131
132 // Enter any new links at the parent's previous position.
133 var linkEnter = link.enter().insert('path', "g")
134     .attr("class", "link")
135     .attr('d', function(d){
136       var o = {x: source.x0, y: source.y0};
137       return diagonal(o, o)
138     });
139
140 // UPDATE
141 var linkUpdate = linkEnter.merge(link);
142
143 // Transition back to the parent element position
144 linkUpdate.transition()
145     .duration(duration)
146     .attr('d', function(d){ return diagonal(d, d.parent) });
147
148 // Remove any exiting links
149 var linkExit = link.exit().transition()
150     .duration(duration)
151     .attr('d', function(d) {
152       var o = {x: source.x, y: source.y};
153       return diagonal(o, o)
154     })
155     .remove();
```

```
156
157   // Store the old positions for transition.
158   nodes.forEach(function(d){
159     d.x0 = d.x;
160     d.y0 = d.y;
161   });
162
163   // Creates a curved (diagonal) path from parent to the child
        nodes
164   function diagonal(s, d) {
165
166     path = `M ${s.y} ${s.x}
167             C ${(s.y + d.y) / 2} ${s.x},
168               ${(s.y + d.y) / 2} ${d.x},
169               ${d.y} ${d.x}`;
170
171     return path
172   }
173
174   // Toggle children on click.
175   function click(d) {
176     if (d.children) {
177         d._children = d.children;
178         d.children = null;
179       } else {
180         d.children = d._children;
181         d._children = null;
182       }
183     update(d);
184   }
185   }
```

2.3.3 Publication and Citation Charts

processing What's more, what we have done is some statistical work. Like other scholar search engine, we count an authors papers and citations by year, and show this information in a histogram so that the user can know the trends in the authors publication of papers. We used chart.js to draw it. Beyond that, we also calculate an authors h-index and i-10-index to provide more detailed information.

```
1
2 //Citation charts
3
4         var citationByYear=<?php echo json_encode($citationByYear);
      ?>;
5         var data=new Array();
6         var labels=new Array();
7         for(var year in citationByYear)
8         {
9           labels.push(year);
10          data.push(citationByYear[year]);
11        }
12
13        var legend = document.getElementById('legend');
14        var ctx=document.getElementById("bar");
15        var myBarChart = new Chart(ctx, {
16            type: 'bar',
17            data: {
```

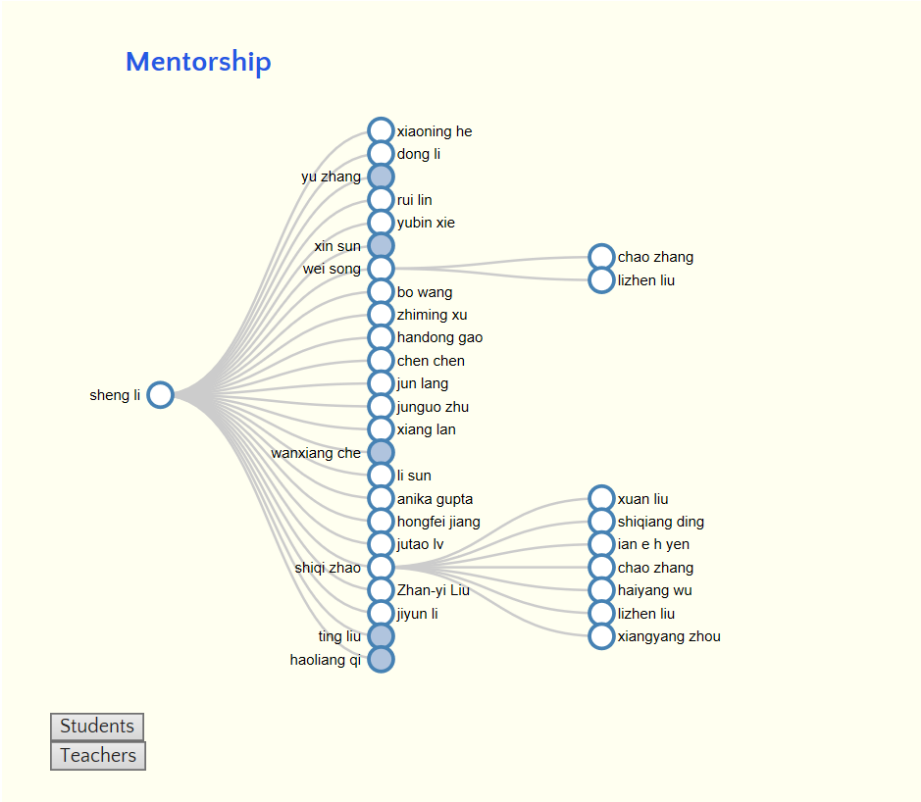


Figure 10: Mentorship

```
                labels:labels,
                datasets:[{
                    label: 'citation',
                    data:data,
                    backgroundColor:"rgba(  0,255,127,0.5)",
                    borderColor:"rgba(151,187,205,0.8)",
                    borderWidth:1,
                }]

            },

            // options: configs
        });
        // legend.innerHTML = data;

//Publication Charts

        var publishes=<?php echo json_encode($publishes); ?>;
        var data=new Array();
        var labels=new Array();
        for(var year in publishes)
        {
          labels.push(year);
          data.push(publishes[year]);
        }
        var legend = document.getElementById('legend2');
```

```
        var ctx=document.getElementById("bar2");
        var myBarChart = new Chart(ctx, {
            type: 'bar',
            data: {
                labels:labels,
                datasets:[{
                    label: 'papers',
                    data:data,
                    backgroundColor:"rgba(  0,255,127,0.5)",
                    borderColor:"rgba(151,187,205,0.8)",
                    borderWidth:1,
                }]

            },

            // options: configs
        });
        // legend.innerHTML = data;
```

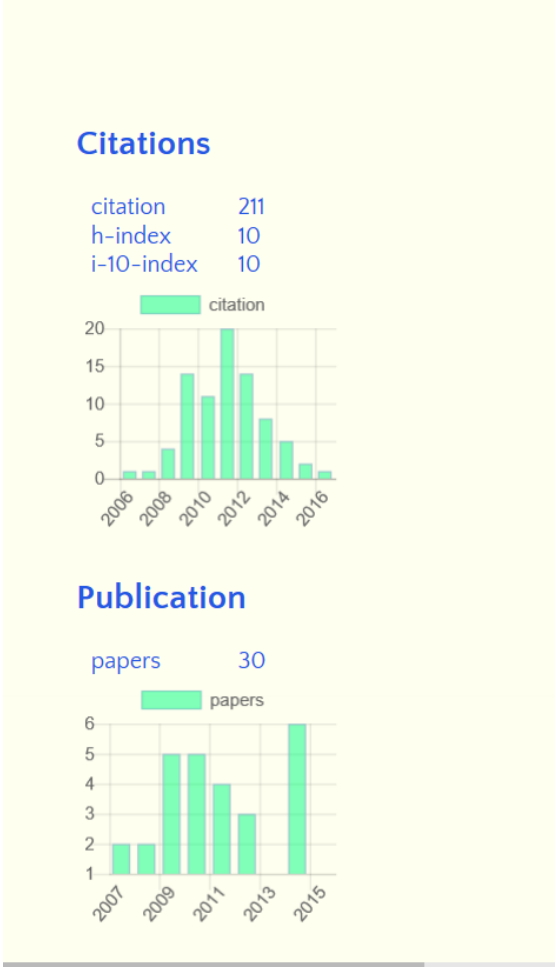


Figure 11: Publication and Citation Charts