The Report of the Project

14 组 沈贯 (324) 李晓 (293) 戴昊悦 (288) 杨子超 (330)

2018 年 6 月 23 日

目录

1 The Task of the Project

1.1 The primary and compulsory tasks

1. To add search for paper title and conference in the home page and search result pages.
2. To build and create as well as modify paper and conference pages in the website.
3. When each page displays information, if there are more information items displayed (such as search results, list of papers, etc.), 10 items are displayed per page and the page function is added.

1.2 The advanced and recommended tasks

1. Recommendations (such as the recommendation of co authors, CO quotes, etc.) can be recommended in the paper page, and the paper can be recommended by the scholar's paper on the scholar's page.
2. The search results are no longer based on simple MySQL queries, and can be implemented with Solr or elasticsearch.
3. The display of the tree structure of teacher relationship can expand the teacher relationship tree by layer. It is also necessary to consider the situation of a scholar who may have multiple tutors.
4. Website performance acceleration, database optimization, SQL statement optimization, etc.

1.3 The innovated and characteristic tasks

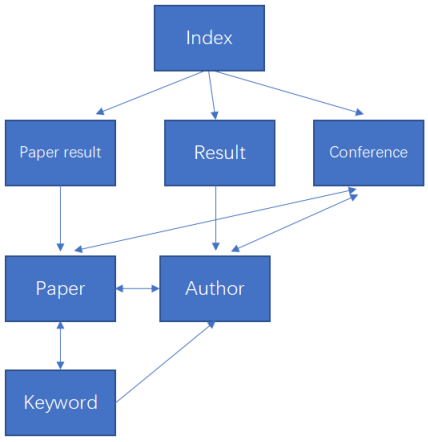
1. Allowing users to sign up and log in and user's information including preferences and visiting history will be recorded.

2. We also design and finish the key word page, you can give likes and hates for this topic and every key word page will recommend the user who is favour of this topic too.

3. The recommendations will be more advanced and intelligent and they are based on user's preferences and visiting history.

2 The basic structure of the website

This is the basic structure of our website. Many ways are designed for users to jump from one page to another. Based on the MVC framework, it is easy to add new pages (views in MVC) into it. Here are some explanations for the pages. Every page except index included a navigation bar which provides the hyper-link to index and the search box to go to the search result. The search results (the second layer in the chart) would return a list of the candidates (papers or authors). There might be a large amount of results. So we designed a simple pagination. And users can click the hyper-link on the page to get more information (go to the specific page of the candidate). And the details would be shown in the following part.



3 The primary and compulsory tasks

3.1 The search for paper title and conference

3.1.1 Analysis

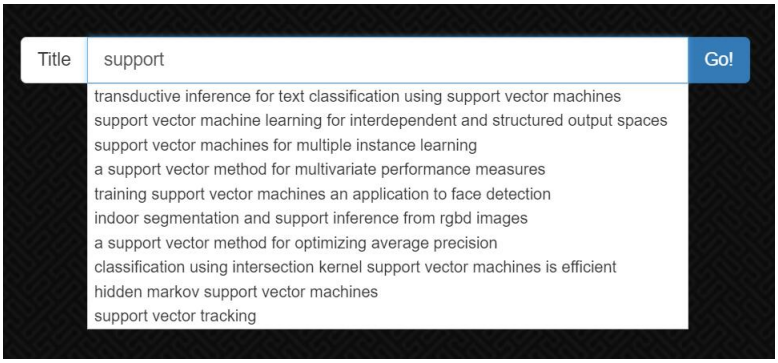
It is a relatively easy and simple task for that we have finished similar tasks before and what we need to do is simply to simulate and slightly modify the precedent source codes to make them adapt to this task.We first expand the original search boxes which just included search for authors and then attaches the hyperlinks and absolute and relative url of the back end for searching for paper title and conferences. Then, we finish writing the source codes for the search box and the implementation of search in the relevant php files including sql sentences and further elasticsearch sentences. After that,we also make some beautification to make the search box look better and striking with the use of bootstrap.And then, we finish the auto-complete for paper title.Due to the fact that the number of conferences is small and we don't finish auto-complete for it.Finally,we decide to use fuzzy search to help users find as many and precise search results as possible.Because the source codes in the back end are a little bit long and not significant.Thus we simply display the source codes for the search boxes.

3.1.2 Implementation

```
1   function switchmethod()
2       {
3           switch (label)
4           {
5               case "Conference": action="author_post/"; label="Author";
6               $("#inputbox").autocomplete({
7               source: "<?=site_url("Search/author_auto_complete/")?>",
8               minLength: 2
9               });
10              break;
11              case "Author": action="paper_post/"; label="Title";
12              $("#inputbox").autocomplete({
13              source: "<?=site_url("Search/paper_auto_complete/")?>",
14              minLength: 2
```

```
                    });
                    break;
                    case "Title": $("#inputbox").autocomplete({source:"<?=site_url("Search/pass/")
                        ?>",minLength:2}); action="conference_post/"; label="Conference"; break;
                }
                $("#btn").html(label);
                $("#search").attr("action","<?=site_url("Search/")?>"+action);
            };
<form class="bs-example bs-example-form" role="form" action="<?=site_url("Search/author_post/")
    ?>" method="post" id="search">
                            <div class="input-group">
                                <span class="input-group-btn">
                                    <button class="btn btn-default" onclick="switchmethod()" id="
                                        btn" type="button">Author</NOtton>
                                </span>
                                <input type="text" class="form-control form-control-lg" placeholder
                                    ="Input anything to search" id="inputbox" name="key">
                                    <span class="input-group-btn">
                                <button class="btn btn-primary" type="submit">Go!</NOtton>
                                </span>
                            </div>
                        </form>
```

3.1.3 Effect



3.2 The conference and paper pages

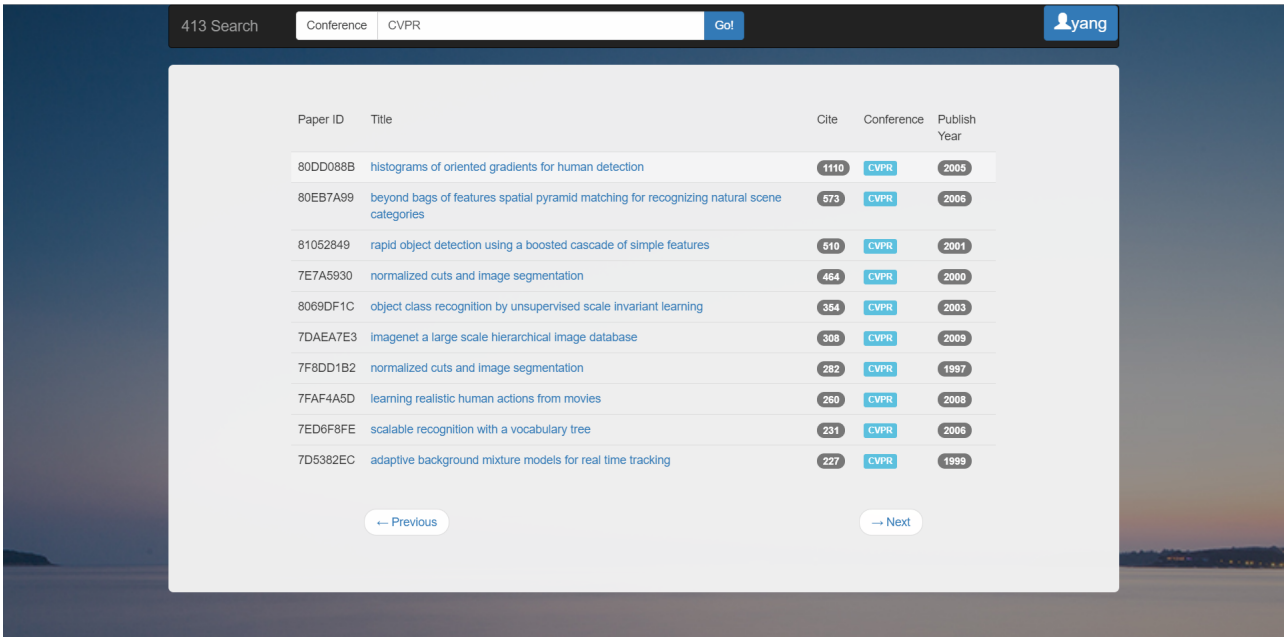
3.2.1 Analysis

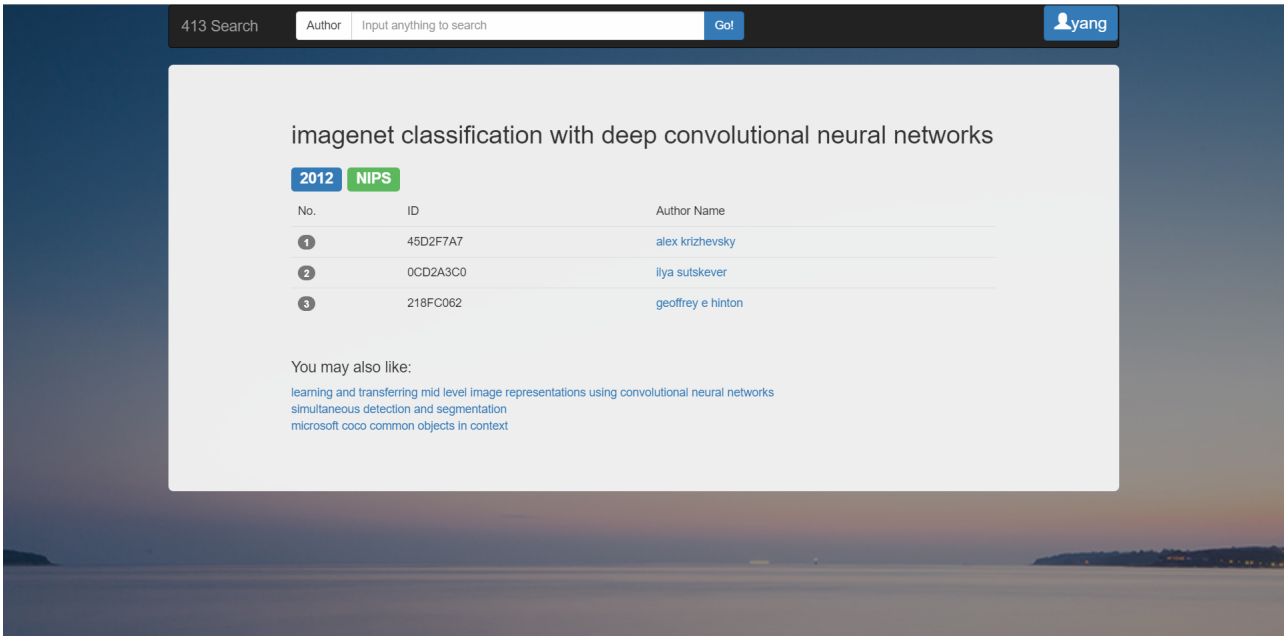
The difficulties of the pages of conference and paper are not how to write the correct source codes because we have written some similar pages, instead,how to decide what to display and how to display? We must decide what will be displayed in the relevant pages due to the fact that there aren't compulsory requirements and we must make the most use of our idea and intelligence to display the valuable and meaningful information for the user.After that we must decide how to display? We must use different regions to display different kinds of information to let user clearly find out the necessary information. We also need to make the outlook of there two pages unified and coordinated with the precedent pages to combine them into a complete and well-organized website.
In the conference page, all the papers of this conference will be displayed and every page will display ten piece of papers.The base of rank is based on the citations,the more, the more preceding.And the PaperID,title and publish year will also be displayed.And hyper links are attached to the title,when clicked,it will jump into the relevant pages.
In the paper title page,all the authors including their AuthorID and author name of this paper will be displayed in the order of author sequence.And it will also display its conference, publish year.What's more, the page will intelligently recommend similar papers.

3.2.2 Implementation

The ordinary source codes for these two pages is of no importance and we don't want to display them.The source codes for recommendation will be displayed in the further section for the advanced and recommended tasks.

3.2.3 Effect


413 Search
Conference
CVPR
Go!
yang
Paper ID
Title
Cite
Conference
Publish Year
80DD088B
histograms of oriented gradients for human detection
1110
CVPR
2005
80EB7A99
beyond bags of features spatial pyramid matching for recognizing natural scene categories
573
CVPR
2006
81052849
rapid object detection using a boosted cascade of simple features
510
CVPR
2001
7E7A5930
normalized cuts and image segmentation
464
CVPR
2000
8069DF1C
object class recognition by unsupervised scale invariant learning
354
CVPR
2003
7DAEA7E3
imagenet a large scale hierarchical image database
308
CVPR
2009
7F8DD1B2
normalized cuts and image segmentation
282
CVPR
1997
7FAF4A5D
learning realistic human actions from movies
260
CVPR
2008
7ED6F8FE
scalable recognition with a vocabulary tree
231
CVPR
2006
7D5382EC
adaptive background mixture models for real time tracking
227
CVPR
1999
← Previous
→ Next


413 Search
Author
Input anything to search
Go!
yang
imagenet classification with deep convolutional neural networks
2012
NIPS
No.
ID
Author Name
1
45D2F7A7
alex krizhevsky
2
0CD2A3C0
ilya sutskever
3
218FC062
geoffrey e hinton
You may also like:
learning and transferring mid level image representations using convolutional neural networks
simultaneous detection and segmentation
microsoft coco common objects in context

3.3 The page turn

3.3.1 Analysis

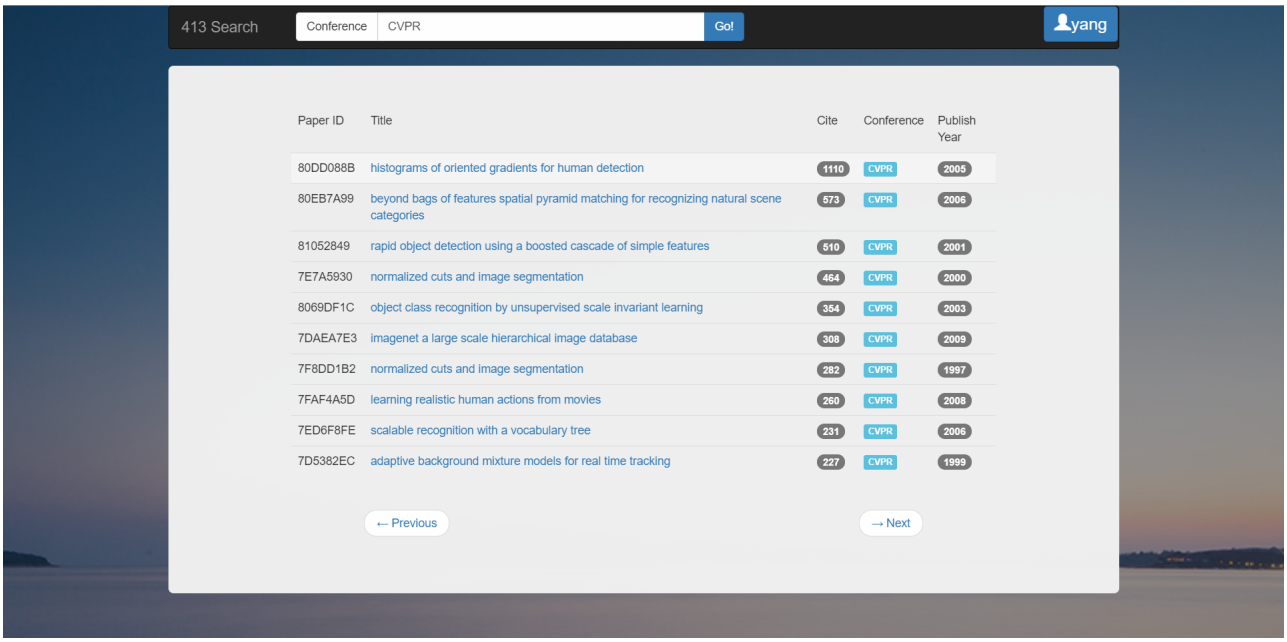
It is a relatively easy and simple task because in lab4, with the use of ajax, we have finished these task. What we need to do is to according to the actual situation of the new conference and paper tile pages, we modify and make a slight change on the source codes and transplant them nearly totally.First,we create two buttons and bind functions with them.And we then write the ajax function and the function used to substitute the pre-existing contents.The source codes for conference pages about page turn will be displayed.And the source codes in the back end is less important and we decided to ignore them.

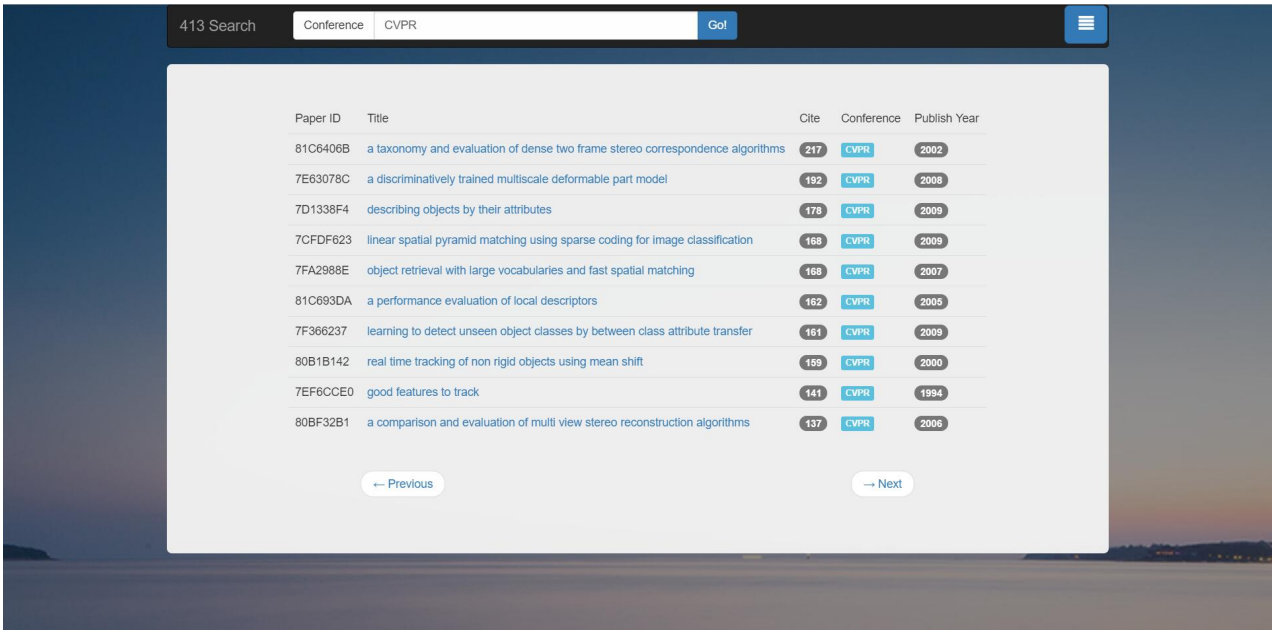
3.3.2 Implementation

```
function multi_author_request(Author_Name,new_Page) {
        if (new_Page==0)
        {
            alert("It has been the first page");
            return;
        }
        if (new_Page><?=$inform["Total_Page"]?>)
        {
            alert("It has been the last page");
            return;
        }
        $.ajax({
            url : "<?=site_url("Search/multi_author_request/")?>",
            dataType : "json",
            type : "post",
            async : true,   //avoid the page collapse
            data : {"Author_Name":Author_Name,"Page":new_Page},
            success : function(data) {
            var html ="<thead><tr><td>Author ID</td><td>Author Name</td><td>Paper Number</td><
                td>Affiliation</td></tr></thead>";
            var prefix = "<?=site_url("Search/single_author_page/")?>";
            Page=new_Page;
            for(var i=0;i<10;i++){
                var row = data[i];
                if (row==undefined) continue;
                html +="<tr><td>"+row["Author_ID"]+"</td><td><a href='"+prefix+row["Author_ID"
                    ]+"'>"+row["Author_Name"]+"</a></td><td><span class=\"badge\">"+row["NUM"
                    ]+"</span></td>";
                if (!row["Affiliation_Name"]) html +="<td><span class='label label-danger'>None
                    </span></td></tr>";
```

```
27                  else html +="<td><span class='label label-info'>"+row["Affiliation_Name"]+"</
                        span></td></tr>";
28                  }
29              $("#Table").html(html);
30              }
31              })
32      }
```

3.3.3 Effect





4 The advanced and recommended tasks

4.1 Recommendations

4.1.1 Analysis for basic principle

This is the most basic way in designing recommendation system: with a specific paper given, we can easily find the papers that share co-authors with it, that cites or be cited by the paper, the papers from predicted teachers or students of the author, from the same affiliation and so on. The fact means that all papers can be connected into a web, with anfractuous relationship between each other, and what we are supposed to do is to dig out the relationship and fetch the papers linked to the given one. Take PHP codes for instance:

4.1.2 Implementation for basic principle

```
$con = mysqli_connect("localhost","","","acemap");
$HisRelated=array();
$sql1 = "SELECT PaperID FROM paper_reference where ReferenceID='{$PaperID}'";
$result1 = mysqli_query($con,$sql1);
while($x=mysqli_fetch_array($result1,MYSQLI_ASSOC)){
    array_push($HisRelated,$x['PaperID']);
};
$sql2 = "SELECT ReferenceID FROM paper_reference where PaperID='{$PaperID}'";
$result2 = mysqli_query($con,$sql2);
while($x=mysqli_fetch_array($result2,MYSQLI_ASSOC)){
    array_push($HisRelated,$x['ReferenceID']);
};
$HisRelated2=array();
foreach ($HisRelated as $one){
    $sql3 = "SELECT Cite,Title FROM papers where PaperID='{$one}'";
    $result3 = mysqli_query($con,$sql3);
    $x=mysqli_fetch_array($result3,MYSQLI_ASSOC);
    array_push($HisRelated2,array('PaperID'=>$one,'Cite'=>$x['Cite'],'Title'=>$x['Title']));
}
if(count($HisRelated2)<=3){
    foreach ($HisRelated2 as $onerec){
        $RecPaperID=$onerec['PaperID'];
        $RecTitle=$onerec['Title'];
        echo"<a href='http://localhost/exercise2/paper.php?PaperID={$RecPaperID}'>{$RecTitle}</
            a>"."<br>";
    }
}else{
    $FinalRec=array();
    while(count($FinalRec)<3){
        $OneBasicRecom=OneBasicRecom($HisRelated2);
```

```
30          if(! in_array($OneBasicRecom,$FinalRec)){
31              array_push($FinalRec,$OneBasicRecom);
32          }
33          }
34      foreach ($FinalRec as $onerec){
35          $RecPaperID=$onerec['PaperID'];
36          $RecTitle=$onerec['Title'];
37          echo"<a href='http://localhost/exercise2/paper.php?PaperID={$RecPaperID}'>{$RecTitle}</
                a>".”<br>”;
38      }
39  }
```

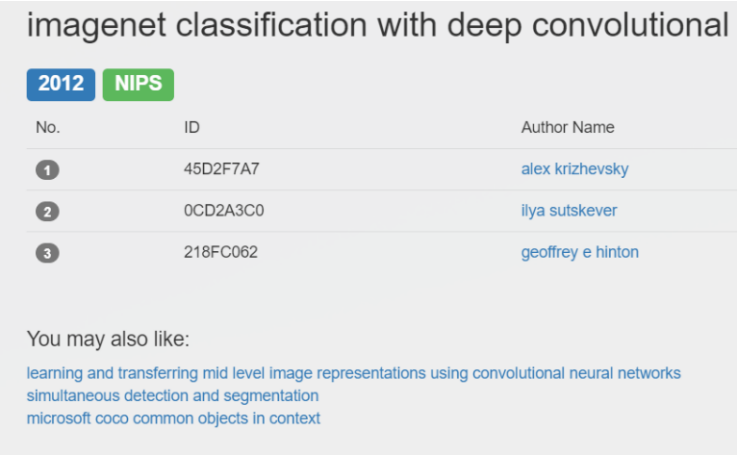
4.1.3 Analysis for weighted random

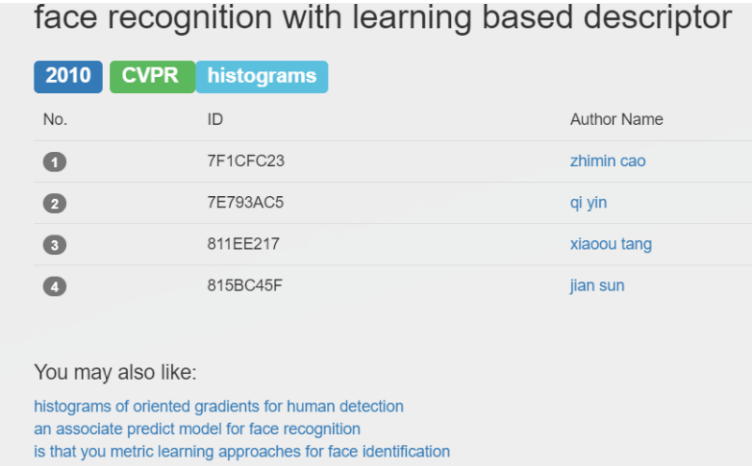
From the codes above, we can find something interesting: what if we just want 3 papers, not all papers in a recommendation time? In fact, just as a commodity product has its ‘popularity’, we also judge a paper’s popularity by the time it’s cited. First we store all the papers dig from database in the candidate list ($HisRelated2 in codes above), and then with the method of ‘Weighted Random’, every paper in the list has the possibility to be recommended, while the more popular a paper is, the more likely it is to be recommended. The PHP function (OneBasicRecom) to accomplish weighted random is as follows:

4.1.4 Implementation for basic principle

```
1   function OneBasicRecom($list) { ('PaperID'=>,'Cite'=>,'Title'=>)
2       $sum = 0;
3       $listPoint = array(0);
4       foreach ($list as $key) {
5           $sum+=$key['Cite'];
6           array_push($listPoint, $sum);
7       }
8       $num = rand(0, $sum);
9       for ($i = 0; $i < count($list); $i++)
10      {
11          if ($num >= $listPoint[$i] && $num < $listPoint[$i+1])
12          {
13              return $list[$i];
14          }
15      }
16      return $list[count($list)-1];
17  }
```

4.1.5 Effect





4.2 Optimization of the Database

4.2.1 Analysis

We spared no effort to accelerate the speed of loading the page. And there are mainly three methods: preprocessing, ajax and Elastic Search.

1. **Preprocessing**:

   This is easy to understand and also used widely. Because the result should be sorted by the paper number or the citation number, it is natural to calculate them before the search and save them in the database. We did an experiment. The speed with preprocessing is at

least 3 times faster than that without preprocessing. However, there is also some "cost" . These data are redundant. If new papers are added into the database, these data should be changed accordingly to maintain the correctness of the result. But we think the speed of the client is more important.

2. **Ajax**:

   Ajax (Asynchronous JavaScript and XML) is a set of web development techniques to create asynchronous web applications. To be more specific, it can get the data from the server asynchronously without interfering the display. By using it, there is no need to display all of the data and it can load the left data whenever needed. We use it in as much as we could to make the speed of loading faster, including the pagination, graph and some not so important data (like the authors of a paper).

3. **Elasticsearch**:

   Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. It uses the inverted index to index the data. And this makes it excel at the tasks like "like %sam%" , which in MySQL is a heavy task of high complexity. Some concepts are similar to MySQL. But after reading about the official document, we find that we couldn' t easily translate the MySQL sentence to Elastic Search. All the operator to Elastic Search is based on JSON data. Elastic Search is far more powerful than we estimated. But we decided to use it in just some scenes it excels at. In some traditional tasks, we still used MySQL. I installed Elastic Search 6.0.1 and the Elastic Search Head which is a tool to manage the database.

   The standard analyzer is used by default for any full-text analyzed

string field. And some functions will use it. By separating the title into several individual words, we could go deeper from searching by matching the string to matching the words. This is based on the fact that we often searched papers by some of the words in its title (keywords). For example, when users want to search for "support vector machine" , certainly Elastic Search and MySQL would give the same results. But what if we forget the "vector" . In MySQL, if you search by "LIKE %support machine%" , you won' t get anything. But if we use analyzer to index the title and use match phrase by setting the slop properly, we could still get the correct results we want.

As for searching by author name, the problem is different. When entering the name, it is most likely to make mistakes in some letters. Like me, when testing the page, I usually search "xiaoou tang" . But I make mistakes frequently by entering "xiaou tang" . So the strategy should be different from searching by title. We use the fuzzy match provided by Elastic Search.

4.2.2 Implementation

The source codes for database:

```
{
  "mappings": {
    "paper": {
      "properties": {
        "title": {
          "type": "text",
          "analyzer": "standard"
        },
        "paper_id": {
          "type": "text"
        },
        "conference": {
          "type": "text"
        },
        "paper_publish_year": {
          "type": "integer"
        },
        "cite": {
```

```
            "type": "integer"}
        }
    }
  }
}
```

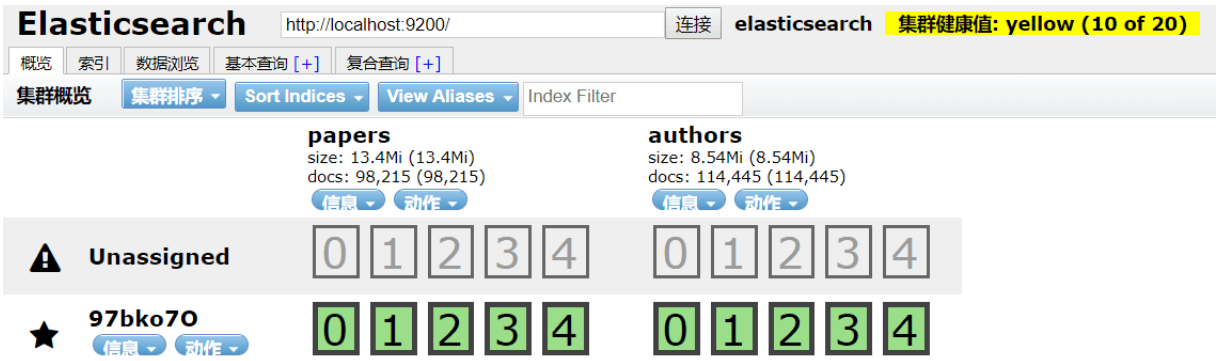
The source codes for search:

```
$query = [
        "index" => "papers",
        "type" => "paper",
        "body" => [
                "query" => ["match_phrase" => ["title"=>["query"=>$title,"slop"=>"3"]]],
                "sort" => ["cite"=>["order"=>"desc"]],
                "size" => $start+ $this->inform_per_page
            ]
        ];
```

The source codes for fuzzy search:

```
$query = [
        "index" => "authors",
        "type" => "author",
        "body" => [
                "query" => ["match" => ["name"=>["query"=>$author_name,"fuzziness"=>"Auto"]]],
                "sort" => ["paper_num"=>["order"=>"desc"]],
                "size" => $start+ $this->inform_per_page
            ]
        ];
```

4.2.3 Effect



4.3 Force Directed Graph

4.3.1 Analysis

For the force directed graph itself, most of the code is from the lab4. And I did a little change, to make dynamical reaction to the users click or

other actions. I also used an extra library called d3-tip, which could help create a more good-looking and practical tip to show the information of every node. To be more specific, when the user put the mouse over the node, then the edge which links the node would be bolder at the same time.

I defined three new events: mouseover, mouseout and dbclick. The mouseover and mouseout provided such function: when the mouse is over one node, edges that link it will be thickened, which could help the user find the cooperators easily. Also, the information of the node will be shown in the tip. The dbclick would be executed when the user double clicks the node. Then the page will jump to his page.

To conclude with, Elastic Search not only brought fast speed but some new functions to make the search friendlier.

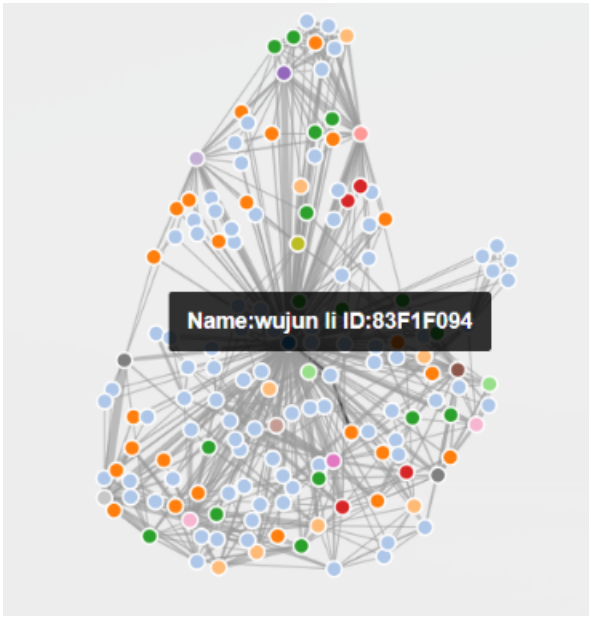
4.3.2 Implementation

```
1
2  var authorprefix = "<?=site_url("Search/single_author_page/")?>";
3  var svg = d3.select("svg"),
4      width = +svg.attr("width"),
5      height = +svg.attr("height");
6
7  var color = d3.scaleOrdinal(d3.schemeCategory20);
8
9  var simulation = d3.forceSimulation()
10     .force("link", d3.forceLink().id(function(d) { return d.id; }).distance(function(d){ return
           d.value}))
11     .force("charge", d3.forceManyBody())
12     .force("center", d3.forceCenter(270,300));
13
14 d3.json("<?=site_url("Search/force_directed_graph/")?><?=$inform["Author_ID"]?>", function(
       error, graph) {
15   if (error) throw error;
16   var link = svg.append("g")
17       .attr("class", "links")
18       .selectAll("line")
19       .data(graph.links)
20       .enter().append("line")
21       .attr("stroke-width", function(d) { return Math.sqrt(d.value); });
22
23   var node = svg.append("g")
24             .attr("class", "nodes")
25             .selectAll("circle")
26             .data(graph.nodes)
```

```
27              .enter().append("circle")
28              .attr("r", 5)
29              .attr("fill", function(d) { return color(d.group); })
30          .on("mouseover",function(d,i){
31              tip.show(d);
32              link.style('stroke-width', function(edge){
33                  if( edge.source === d || edge.target === d ){
34                      return 2*Math.sqrt(edge.value);
35                  }
36              }).style('stroke', function(edge){
37                  if( edge.source === d || edge.target === d ){
38                      return '#000';
39                  }
40              });
41          })
42          .on("mouseout",function(d,i){
43              tip.hide();
44              link.style('stroke-width', function(edge){
45                  if( edge.source === d || edge.target === d ){
46                      return Math.sqrt(edge.value);
47                  }
48              }).style('stroke', function(edge){
49                  if( edge.source === d || edge.target === d ){
50                      return '#ddd';
51                  }
52              });
53          })
54      .on("dblclick",function(d,i){
55          window.open(authorprefix+d.id);
56      })
57          .call(d3.drag()
58              .on("start", dragstarted)
59              .on("drag", dragged)
60              .on("end", dragended));
61      var tip = d3.tip()
62        .attr('class', 'd3-tip')
63        .offset([-10, 0])
64        .html(function(d) {
65          return "Name:"+d.name+"\nID:"+d.id;
66        })
67
68      svg.call(tip);
69
70      simulation
71          .nodes(graph.nodes)
72          .on("tick", ticked);
73
74      simulation.force("link")
75          .links(graph.links);
76      simulation.restart();
77
78      function ticked() {
79        link
80            .attr("x1", function(d) { return d.source.x; })
81            .attr("y1", function(d) { return d.source.y; })
82            .attr("x2", function(d) { return d.target.x; })
83            .attr("y2", function(d) { return d.target.y; });
84
```

```
85      node
86          .attr("cx", function(d) { return d.x; })
87          .attr("cy", function(d) { return d.y; });
88    }
89
90  });
91
92  function dragstarted(d) {
93    if (!d3.event.active) simulation.alphaTarget(0.3).restart();
94    d.fx = d.x;
95    d.fy = d.y;
96  }
97
98  function dragged(d) {
99    d.fx = d3.event.x;
100   d.fy = d3.event.y;
101 }
102
103 function dragended(d) {
104   if (!d3.event.active) simulation.alphaTarget(0);
105   d.fx = null;
106   d.fy = null;
107 }
```

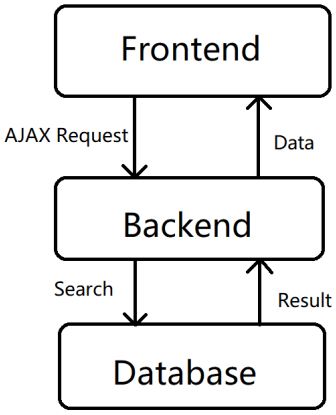
4.3.3 Effect



4.4 Mentoring Tree

4.4.1 Analysis

The basic idea to draw a mentoring tree is simple. Firstly, we get the data of relationship between the scholars, then draw the graph. At first, we use a backend page to searching scholar in our database, translating the format of the data. Secondly, send the data to frontend page by jQuery AJAX function. Finally, using javascript code to draw the tree with the SVG element.



If we show all the students and advisors of a scholar, we can get nearly infinite people——almost everyone has an advisor. For the users, what they are mainly interested in is the status of a scholar. They may want to know the advisor of the scholar, but not advisor of the scholar' s advisor. When it comes to the students of the scholar, they may have an interest in how many students the scholar has, and the students of his (or her) students. So, we only need to present the students tree of a scholar.

If we treat students of the scholar as first layer, the students of his students as second layer, to let our page be clean and succinct, we only show four layers. For further relationships, they have little connection with the scholar we search, so we skip them to make the page clean.

A scholar may have advisor more than one, but in mentoring tree,

a node only has one root. That' s another reason why we only show mentoring tree from the students' direction. When we search scholar A, showing his student B. B may have other advisors, but when we search scholar A, we only present A is B' s advisor. This principle guarantees the completeness of mentoring tree.

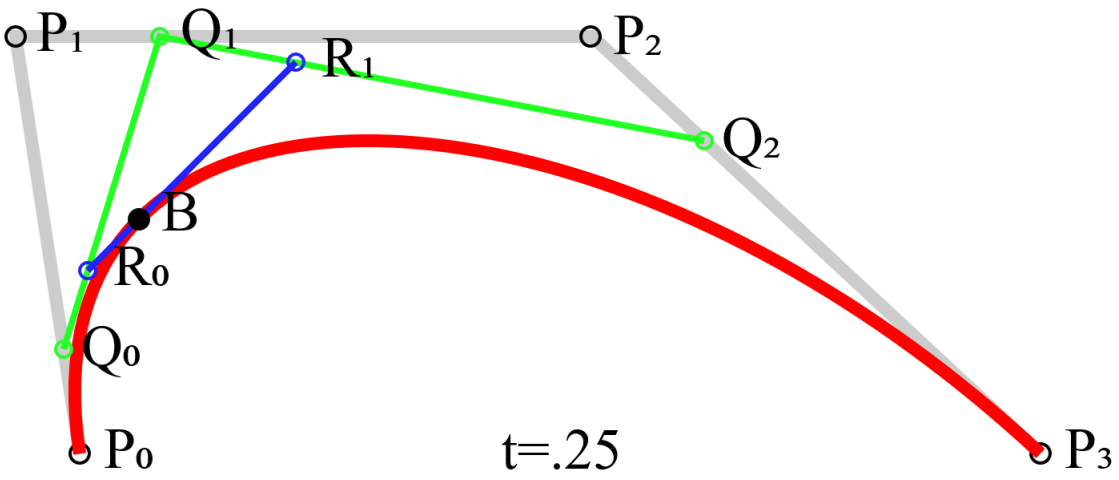
Now we accomplish the backend page. We obtain data of relationship in table "relationship", in which it stores a scholar' s id and his advisor' s id. When it comes to search scholar in database, we take such strategy:

- Search the student' s id whose "teacher" id equal to the scholar' s id. These are the first layer of the mentoring tree. We not only add relationship "scholar->students" to "$data", but also create a box to store these students id.
- Then we turn to the next layer. All the id in box are the root nodes of the next layer. We create a circulate to build each layer. In each circle, we build a new box, searching ids stored in old box as "teacher", adding relationship between them and their students, putting their students into the new box for next circulate. At the same time, if the students name has been in the data set, skip them in the next circulate.
- Finally, return the data in json.

Now we come to finish the frontend part. We using javascript code, so this function could be add into the webpage easily. The html show graph by "SVG" element, which has a kind of elements: path. By path element, we can draw Bézier curve, we use which to connect two node in our mentoring tree.

Bézier curve is a parametric curve, composed of set of control points and line between them. For Cubic Bézier curves, it has four point: start

point P1, end point P2 an d two medium points P3、P4. Move two medium points, the shape of curve will change.



D3 document provide a useful function “d3.startify()” to construct a root node from tabular data. It will return data in hierarchical structure. Then we have two choice: use tree structure or cluster structure. We choose tree structure.

Besides, an important function we hope to accomplish is that when we click a point, the children part of this node will disappear, and if we click again, the hidden part will come into light. To accomplish this, we write an “redraw” function, whenever we click the node, modify the data and the whole graph will redraw. Here comes the detail.

4.4.2 Implementation

Prepare function and parameter we will use.

```
1
2  var stratify = d3.stratify()
3      .id(function(d) {
4          return d.name;
5      })
6      .parentId(function(d) {
7          return d.parent;
8      });
```

This function translates the data format from tabular to hierarchical.

```
function connector(d) {
    return "M" + d.y + "," + d.x +
        "C" + (d.y + d.parent.y) / 2 + "," + d.x +
        " " + (d.y + d.parent.y) / 2 + "," + d.parent.x +
        " " + d.parent.y + "," + d.parent.x;}
```

This function is the Bézier curves which connect to nodes.

```
function collapse(d) {
    if (d.children) {
        d._children = d.children;
        d._children.forEach(collapse);
        d.children = null;
    }
}
```

When we open the page refresh the page, this function collapse all the mentoring tree branch.

Other function we use should be define in the next part.

Use AJAX get data, calculate the data and translate them into tree structure.

```
d3.json("relationship.php?id="+id,function (data) {
        var root = stratify(data);

        root.each(function(d) {
            d.name = d.id; //transferring name to a name variable
            d.id = i; //Assigning numerical Ids
            i++;
        });
                root.x0 = 540;
                root.y0 = 300;
        root.children.forEach(collapse);  //collapse the tree initially
other code……
});
```

Now we define the redraw function. Get the tree structure.

```
function update(source) {

    var nodes = tree(root).descendants(),
        links = nodes.slice(1);
```

The code above this line is overlook the link from the root to the root itself.

Then update the node. When the redraw function been called, the data has changed.

```
var node = svg.selectAll("g.node")
    .data(nodes, function(d) { return d.id || (d.id = ++i); });

var nodeEnter = node.enter().append("g")
    .attr("class", "node")
    .attr("transform", function(d) { return "translate(" + source.y0 + "," + source.x0 + ")";
        })
    .on("click", click);
```

"nodeEnter" are the nodes which collapsed before but we extend it this time. And the function "click" is defined as following:

```
        function click(d) {
    if (d.children) {
        d._children = d.children;
        d.children = null;
    } else {
        d.children = d._children;
        d._children = null;
    }
    update(d);
}
```

So if we click the node, hide the data to variable "_children", and redraw the SVG element.

```
var nodeExit = node.exit().transition()
    .duration(duration)
    .attr("transform", function(d) { return "translate(" + source.y + "," + source.x + ")"; })
    .remove();
```

delete the nodes which has been hidden. D3 provide function to achieve gradual change, set the variable "duration" to 500 millisecond.

Then append circle,text to the node, set the color, set the parameter of the text. Now draw the curve.

```
var link = svg.selectAll("path.link")
    .data(links, function(link) { var id = link.id + '->' + link.parent.id; return id; });
link.transition()
    .duration(duration)
    .attr("d", connector);
var linkEnter = link.enter().insert("path", "g")
    .attr("class", "link")
    .attr("d", function(d) {
        var o = {x: source.x0, y: source.y0, parent:{x: source.x0, y: source.y0}};
        return connector(o);
    });
```

Transition links to their new position. Enter any new links at the parent's previous position.

```
link.exit().transition()
    .duration(duration)
    .attr("d", function(d) {
        var o = {x: source.x, y: source.y, parent:{x: source.x, y: source.y}};
        return connector(o);
    })
    .remove();
```

Transition exiting nodes to the parent's new position. Then delete other link. That' s the basic idea of the frontend page.

4.5 Difficulty

There exist some difficult problem during our work. The biggest problem we met in this part is how to implement the function of redraw. First we have to define redraw function in "d3.json()" part, and the function "click" also must define in this part. We have made mistake that the function define in different part so they can' t work together. Then how to update the condition of nodes also become difficulty. Finally, setting the parameter to draw the Bézier curve.

5 The innovated and characteristic tasks

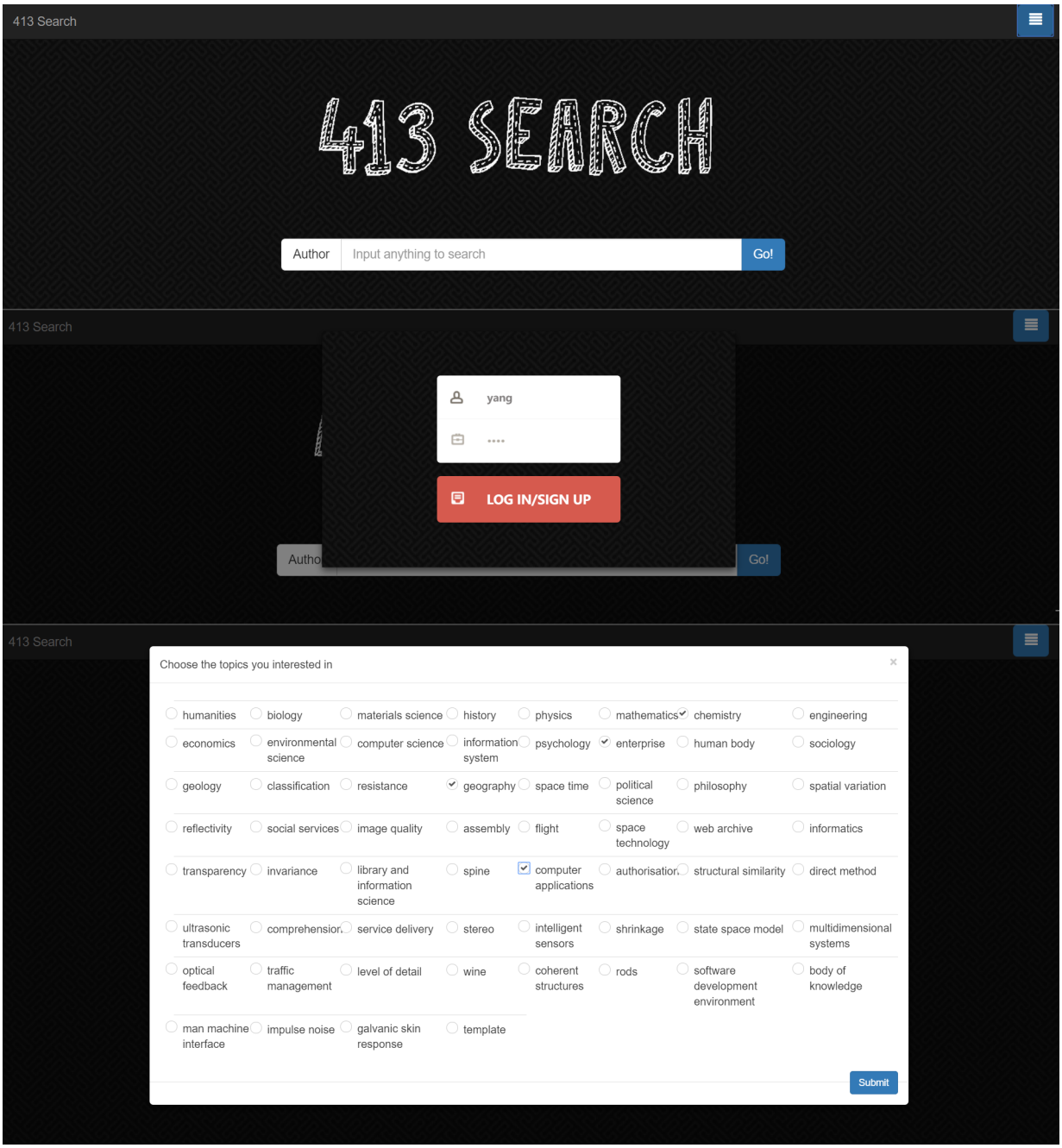
5.1 The appearance and interaction of home page

We used bootstrap to develop the appearance of the pages. Bootstrap is a free and open-source front-end framework (library) for designing websites and web applications. And we also used a little animate(animate.css) to develop a dynamic effect to make the interaction friendlier to the users. The concepts of container and column make it easier to set type.

The index is the start of the website, for the sake of which it only

provides the simplest functions: search and log in (or sign up). The input box could switch the mode to search by clicking the left button: authors, papers or conference. And the blue button upper right would call a modal for the user to log in or sign up. It would detect automatically if the account exists. And if the account is not found, the user will need to choose the topics(keywords) he(she) is interested in. This would relate to the function of recommendation. When the go button is clicked , the key the users input will be posted to the controller and the controller would get the data needed from the model for the new page. Because we come from the same dormitory 413, thus we name our website 413 search.And we also use animate.css to finish the dynamic effect and it is rather cute and interesting.And it is hard to display 3 photos together in latex due to the fact that the location of photo is not fixed.Thus we cat and paste the three photos together and make 3 photos become one photo.

5.1.1 Effect

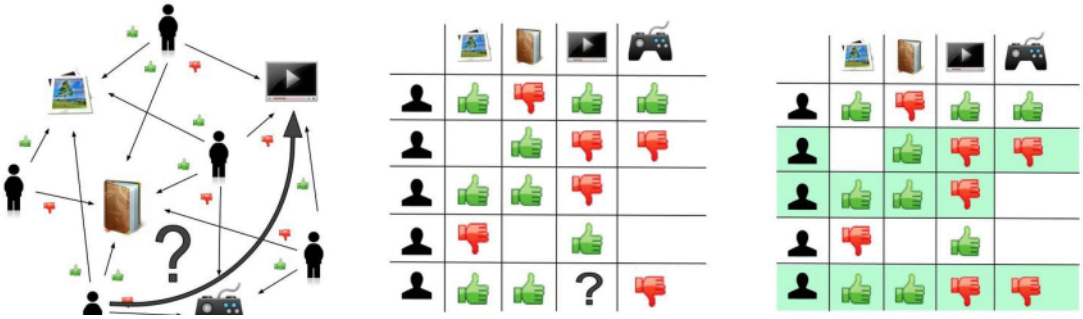

413 Search
413 SEARCH
Author
Input anything to search
Go!
413 Search
yang
LOG IN/SIGN UP
Go!
413 Search
Choose the topics you interested in
humanities
biology
materials science
history
physics
mathematics
chemistry
engineering
economics
environmental science
computer science
information system
psychology
enterprise
human body
sociology
geology
classification
resistance
geography
space time
political science
philosophy
spatial variation
reflectivity
social services
image quality
assembly
flight
space technology
web archive
informatics
transparency
invariance
library and information science
spine
computer applications
authorisation
structural similarity
direct method
ultrasonic transducers
comprehension
service delivery
stereo
intelligent sensors
shrinkage
state space model
multidimensional systems
optical feedback
traffic management
level of detail
wine
coherent structures
rods
software development environment
body of knowledge
man machine interface
impulse noise
galvanic skin response
template
Submit

5.2 Collaborative filtering

5.2.1 Analysis and Implementation

1. **Basic Principles**:

   Collaborative Filtering is a recommendation approach widely used in today' s apps and websites, like Taobao, music apps, Facebook and so on. The growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering. Collaborative filtering is one of the techniques used for dealing with this problem. The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis. Collaborative filtering algorithms often require: 1) users' active participation, 2) an easy way to represent users' interests, and 3) algorithms that are able to match people with similar interests.



   An example of predicting of the user's rating using collaborative filtering. At first, people rate different items (like videos, images, games). After that, the system is making predictions about user's rating for an item, which the user hasn't rated yet. These predictions are built upon the existing ratings of other users, who have

similar ratings with the active user. For instance, in our case the system has made a prediction, that the active user won't like the video.

Typically, the workflow of a collaborative filtering system is:

1) A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.

2) The system matches this user's ratings against other users' and finds the people with most "similar" tastes.

3) With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item)

And there are two approach in CF: the user-based one, and the item based one. The difference lies on which is used as axises, and which as vectors.

2. **User System Build**:

   Apparently the CF is based on user system. So I designed a user system, in which visitors should first sign up an account and then browse the websites with logged-in status. With the 'session' method, we collect users' behavior towards different papers, and correspondingly, record and edit his preferences degree in the information about him stored in database.

3. **How to define 'similarity' ?**:

   From the exposition above, we have get a relatively clear impression on how CF works. Then comes the question: how to define, or how to quantize a users behavior? We assume different degrees, like viewing a paper for more than 30s adds +1; 'like' a paper by

clicking the ‘like’ button adds +3; while ‘dislike’ adds -1. Or if the user google the paper soon after he views it, we add positive to his degree of this paper. However users’ viewing track to other websites isn’ t available now, in the principle for safety. The PHP codes to edit one’ s preference are as follows:

```
function KeywordsArrayEdit($userid,$KeywordsArray,$degree){
    $con = mysqli_connect("localhost","","","acemap");
    $sql = "set names utf8";
    mysqli_query($con,$sql);
    $LevelArray=LevelArray($KeywordsArray);
    $sql="SELECT Level0,Level1,Level2,Leveliso FROM user WHERE id={$userid}";
    $result = mysqli_query($con,$sql);
    $x=mysqli_fetch_array($result,MYSQLI_ASSOC);


    $Level0array=explode(',', $x['Level0']);
    foreach ($LevelArray[0] as $x0){
        $Level0array[$x0]+=$degree;
    }

    $Level1array=explode(',', $x['Level1']);
    foreach ($LevelArray[1] as $x1){
        $Level1array[$x1]+=$degree;
    }

    $Level2array=explode(',', $x['Level2']);
    foreach ($LevelArray[2] as $x2){
        $Level2array[$x2]+=$degree;
    }

    $Levelisoarray=explode(',', $x['Leveliso']);
    foreach ($LevelArray[-1] as $xiso){
        $Levelisoarray[$xiso]+=$degree;
    }


    $newstr0=implode(',',$Level0array);
    $newstr1=implode(',',$Level1array);
    $newstr2=implode(',',$Level2array);
    $newstriso=implode(',',$Levelisoarray);
    $sql="UPDATE user SET Level0='{$newstr0}',Level1='{$newstr1}',Level2='{$newstr2}',
        Leveliso='{$newstriso}' WHERE id={$userid}";
    mysqli_query($con,$sql);
}
```

Some part of the codes (like level) will be illustrated in part3. Since we have edit function, we can record a user’s preference in database. Then how tocalculate similarity between different users? The principle is cosine formula: we regard different papers as different axises,

then that' s an n-dimension space. A user' s preference can be demonstrated as a vector in the n-dimension space, and then the similarity between two users can be demonstrated as intersection angle between the two vectors. For preprocessing of vectors:

$r_{u,i} = \text{aggr}_{u' \in U} r_{u',i}$ where 'U' denotes the set of top 'N' users that are most similar to user 'u' who rated item 'i'. Some examples of the aggregation function includes:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

$$r_{u,i} = k \sum_{u' \in U} \text{simil}(u, u') r_{u',i}$$

where k is a normalizing factor defined as $k = 1 / \sum_{u' \in U} |\text{simil}(u, u')|$, and

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \text{simil}(u, u')(r_{u',i} - \bar{r_{u'}})$$

The cosine-based approach defines the cosine-similarity between two users x and y as:

$$\text{simil}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

```php
function getMod($arr){//传入一个数组，返回其模长
    $strModDouble = 0;
    foreach($arr as $val){
        $strModDouble += $val * $val;
    }
    $strMod = sqrt($strModDouble);
    if($strMod==0){return 1;}//防止除数为0
    return $strMod;
}

function getCosine($arrMark, $arrAnaly){//标记向量和分析向量
    $Vector = 0;
    for($i = 0; $i < count($arrMark); $i++){
        $MarkVal = intval($arrMark[$i]);
        $AnalyVal = intval($arrAnaly[$i]);
        $Vector += $MarkVal * $AnalyVal;
        $ModProduct=getMod($arrMark)*getMod($arrAnaly);
    }
    $Cosine = $Vector/$ModProduct;
    return $Cosine;
}
```

4. **Difficulties and Solutions**:

(a) **Cold Setup**:

From the basic ideas above, we can see that if a user has just signed up and merely have any preferences or viewing tracks, it' s difficult to give recommendation to him. So for the newcomers, we combine with the part1 based on Relevance Popularity, and let a user chooses fields he likes in the signing-up process.

(b) **Diversity and the Long Tail**:

Collaborative filters are expected to increase diversity because they help us discover new products. Some algorithms, however, may unintentionally do the opposite. Because collaborative filters recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products, akin to positive feedback. This bias toward popularity can prevent what are otherwise better consumer-product matches. A Wharton study details this phenomenon along with several ideas that may promote diversity and the "long tail." Several collaborative filtering algorithms have been developed to promote diversity and the "long tail" by recommending novel, unexpected, and serendipitous items, which means lots of items may remain deserted, and fails to be recommended as a vicious circle. For solution, random may help but accuracy may be sacrificed. In TURC2018, I attended a lecture given by CEO of Kuaishou. In the lecture he stated that Kuaishou develops unique algorism to remit long tail problem so that common users can get viewers and fans too. I' ll make deeper acquaintance to that later.

(c) **Data Sparsity**:

In practice, many commercial recommender systems are based on large datasets. As a result, the user-item matrix used for collaborative filtering could be extremely large and sparse, which brings about the challenges in the performances of the recommendation. Now we have nearly 100,000 papers in database, that' s quite big for operation, and what' s worse is that, with the higher dimension, intersections between different users all level off to 0, since users won' t view so many papers. In mathematic matrix factorization may help, but I think the most effective way is part3: Content-Based.

5.3 Content-Based

5.3.1 Analysis and Implementation

1. **Basic Principles**:

   One of the biggest problem in CF is data sparsity. Optimization design for the problem is dimensionality reduction: dimension isn' t one-to-one linked to specific items, but to features extracted and classified from items. This is why in reality industrial circles, CF is always used together with CB. Main process of CB can be divided into 3 parts: 1. Item Representation: For each item extract some features (that is, the item's content) to represent this item; 2. Profile Learning: Using a feature data of a user's favorite (and disliked) item in the past to learn the user's preference profile; 3. Recommendation Generation:By comparing the characteristics of the user profiles and candidate items obtained in the previous step, the user is recommended a set of the most relevant items. For papers, the tags (or features) we choose, is keywords.

2. **Keywords Extraction**:

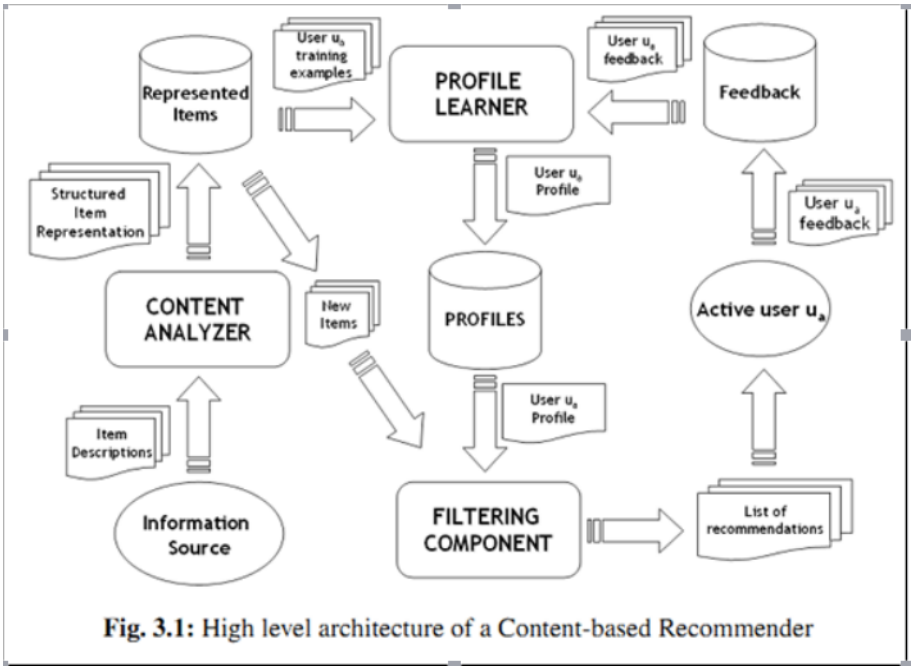


Fig. 3.1: High level architecture of a Content-based Recommender

First I should express my gratitude to Dr.Ja for his providing me with keywords to each paper. However I found it can't meet our requirements for the two reasons below: 1)There are still too many dimensions (keywords) 2)I don't know relationship between keywords. Why is that important? For example, if userA is interested in English Premier League, while userB is interested in German Bundesliga, FIFA should be recommended to both of them, in spite the two keywords are definitely disparate. That's because the two topics both belong to a father topic: football. So we also need topic relationships, that is, Topic Clustering.

My original idea for Topic Clustering is like this: first use web crawler to fetch topic description of topics on web, and then use natural language analysis to build an N-gram language model, finally use TensorFlow to build topic system.

Soon after I found it unnecessary, since a complete topic web is already in Acemap's database. For one topic, its related topics can be shown, and I only need to use web crawler to fetch all these datas. Python codes are as follows:

```
import urllib.request
import pymysql
from bs4 import BeautifulSoup
db = pymysql.connect("localhost","","","acemap" )
cursor = db.cursor()

with open('/Users/markdana/Desktop/keys2.txt', 'r') as f:
    for line in f.readlines():
        try:
            a_key_id=(line.strip()).split('\t')[0]
            file=urllib.request.urlopen("http://acemap.sjtu.edu.cn/topic/topicpage?
                topicID=%s"%(a_key_id))
            data=str(file.read())
            index1=data.find('Parent Topic:')
            index2=data.find('Child Topic')
            rawdata=data[index2:]
            index3=rawdata.find('<br>')
            parent=data[index1:index2]
            child=rawdata[:index3]


            soup1 = BeautifulSoup(parent, features='lxml')
            soup2 = BeautifulSoup(child, features='lxml')

            for node in soup1.find_all('a'):
                nodestr=str(node)
                index1=nodestr.find('topicID=')
                index2=nodestr[index1:].find('\\')
                a_parent=nodestr[index1:][8:index2]
                print(a_parent)
                sql="SELECT*FROM keyword_matrix where Parent='%s'and Child='%s'"%(
                    a_parent,a_key_id)
                cursor.execute(sql)
                result=list(cursor.fetchall())
                if len(result)==0:
                    sql = "INSERT INTO keyword_matrix(Parent,Child)VALUES('%s','%s')" % (
                        a_parent,a_key_id)
                    try:
                        cursor.execute(sql)
                        db.commit()
                    except:
                        db.rollback()

            for node in soup2.find_all('a'):
                nodestr=str(node)
                index1=nodestr.find('topicID=')
                index2=nodestr[index1:].find('\\')
                a_child=nodestr[index1:][8:index2]
                print(a_child)
                sql="SELECT*FROM keyword_matrix where Parent='%s'and Child='%s'"%(
                    a_key_id,a_child)
                cursor.execute(sql)
                result=list(cursor.fetchall())
                if len(result)==0:
                    sql = "INSERT INTO keyword_matrix(Parent,Child)VALUES('%s','%s')" % (
                        a_key_id,a_child)
                    try:
                        cursor.execute(sql)
```

```
                            db.commit()
                        except:
                            db.rollback()

        except:
            print('11111111111111')
```

3. **Define 'Level' Sort**:

   From 3.2, we've already constructed a matrix which shows exactly, a topic's parents and children. Then how can we sort and locate those topics from relationship between them?I define 'Level' for topics, just like 'depth' in graph theory: Level=0: the topic doesn't have a parent. Fundamental topics like biology, physics; Level=1: max distance from fundamental topic point is 1, which means for it there exists a parent whose Level=0; Clearly Level is defined in recursion.

   And Level=-1: Because the database we used for project is far smaller than that of Acemap, some data we craw from web isn't in our database. They are 'isolated points', which means it has parent but the parent doesn't exist in database; or it's parent can be found in database, but also isolated points. For convenience to define level, we assume all the isolated pints are in level -1. The python codes are as follows:

```
import pymysql
db = pymysql.connect("localhost","","","acemap" )
cursor = db.cursor()
def order(A):
    orders=[]
    sql="SELECT Parent FROM keyword_matrix WHERE Child='%s'"%(A)
    cursor.execute(sql)
    tmp=cursor.fetchall()
    for x in list(tmp):
        x_keyid=x[0]
        #print("pp: "+x_keyid)
        sql1="SELECT Level FROM keywords WHERE KeywordID='%s'"%(x_keyid)
        try:
            cursor.execute(sql1)
            parent_order=list(cursor.fetchall())[0][0]
            #print(x_keyid+" ORDER= "+str(parent_order))
        except:
            #print(x_keyid+" ORDER= NONE")
```

```
19                continue
20            if parent_order!=-1:
21                orders.append(parent_order+1)
22            else:
23                x_keyid_order=order(x_keyid)
24
25                sql2 = "UPDATE keywords SET Level='%d' WHERE KeywordID='%s'"%(x_keyid_order,
                      x_keyid)
26                cursor.execute(sql2)
27                db.commit()
28
29                orders.append(x_keyid_order+1)
30
31        if len(list(tmp))==0:
32            return 0
33        try:
34            print(orders)
35            return max(orders)
36        except:
37            return -1
38
39    with open('/Users/markdana/Desktop/keys2.txt', 'r') as f:
40        for line in f.readlines():
41
42            try:
43                a_key_id=(line.strip()).split('\t')[0]
44                int=order(a_key_id)
45                print(int)
46                sql = "UPDATE keywords SET Level='%d' WHERE KeywordID='%s'"%(int,a_key_id)
47                try:
48                    cursor.execute(sql)
49                    db.commit()
50                except:
51                    print("2222222222222")
52                    db.rollback()
53            except:
54                print('11111111111')
55
56
57
58    sql3 = "SELECT KeywordID FROM keywords WHERE Level=-1"
59    cursor.execute(sql3)
60    result=list(cursor.fetchall())
61
62    for line in result:
63        a_key_id=line[0]
64        sql4="SELECT Parent FROM keyword_matrix WHERE Child='%s'"%(a_key_id)
65        cursor.execute(sql4)
66        result=list(cursor.fetchall())
67
68        print(a_key_id)
69        print(result)
70
71        reslist=[]
72        for x in result:
73            id=x[0]
74            sql5="SELECT Level FROM keywords WHERE KeywordID='%s'"%(id)
75            cursor.execute(sql5)
```

```
76          res=list(cursor.fetchall())
77          try:
78              reslist.append(res[0][0])
79          except:
80              reslist.append(-2)
81      print(reslist)
```

4. **Store User' s Preferences**:

   From 3.3, every topic has its own Level and can be located by LevelNo (sorted by TotalAmountOfPapers).Then how to store degrees of different users? MySQL can' t store array, and if I create thousands of columns, it' s quite unnecessary. So I decided to use a string to store users' preferences. For example, there are 96 topics with level 0, then create a column level0, and it stores a string with 96 numbers split with ',' , like this: 2,2,2,2,-1,-2,1,0,0,0,-3,0,0,0,0,-1,0,0,0,0,0,-2,⋯.

5. **Further thoughts**:

   From the whole process of building recommendation system, I have got plain understand of different algorithms. However two main directions need to be improved: 1.Arithmetic Speed. With the user amount increasing, traverse all users and find the most similar may cost lots of time. We can calculate all the time in end back (not just when loading the page); and we can filtrate users into different groups in order. 2.Recommendation Efficiency. We can add more comprehensive functions to the edit module, so that the description of a user can be more accurate. What' s more, we may develop academic social network based on the users, and to stimulate communication between researchers. That' s quite a nice idea.