Lab 3 Web Development
EE 101

https://acemap.lifanz.cn/

Lifan Zhang
Jiahong Li
Kaipeng Zeng
Yanming Liu

2019.5.15

Contents

The aim of our project *PaperHub* is to show the visitors the data about the papers, authors, affiliations and other academic units, so as the basis of the project, we need to do some data processing first. First we want to score the authors according to the citations of his or her papers, and his or her rank of authors in the papers. The maximum number of authers is 57 in the data provided by the teaching assistant of EE101, so we design a function to weight the academic achievement of the author. Then we calculate the score of each author and affiliation, and write it into two new tables in our MySQL database.

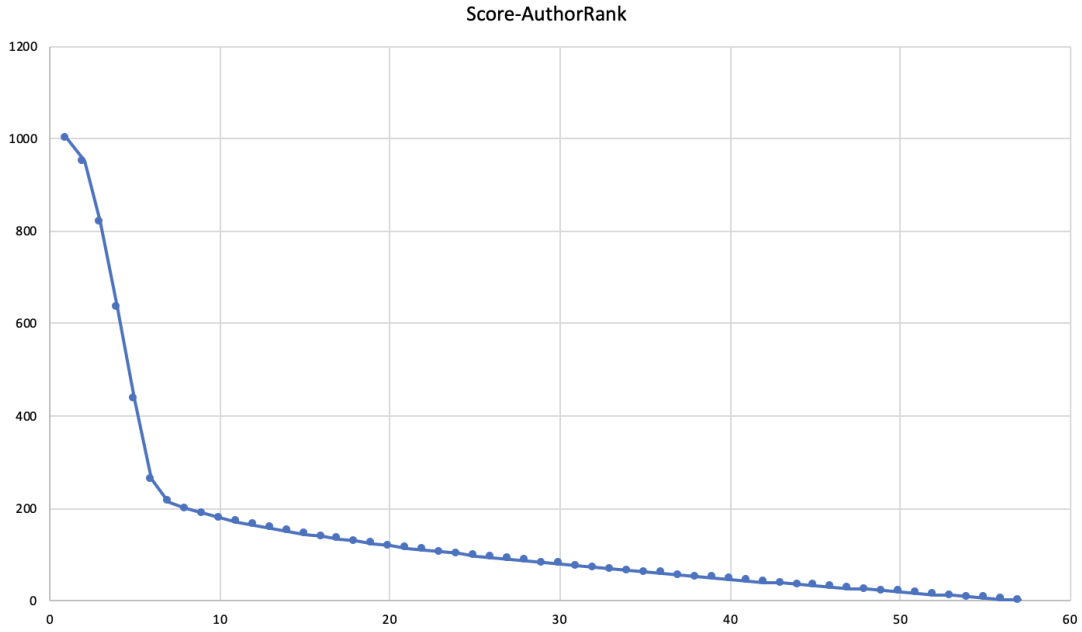


Figure 1: The score range from 2 to 1000

1 Website Deployment

First we buy a domain and an ECS Server from Aliyun and install Ubuntu 18.04. Then we use apt command to install python, MySQL, Nginx, PHP and other essential software. After uploading our website to the server through SFTP, we add a server into the configuration of Nginx to let it listen port 80 on the domain acemap.lifanz.cn. To make our website legal, we registered it to MIIT and the public security department, to get an ICP registration number and a public security registration number. Finally, we use certbot the enable SSL connection throught port 443.

2 Homepage

In the homepage, we want to realize two functions. The first one is a general searching input box. In the box, the user can search everything he want, and the webpage will guess what the user want to search and give the user some hints in a dropdown list. The second one is an advanced search form, where the user can specify the type of data he want to get. We use bootstrap and a model(https://github.com/BlackrockDigital/startbootstrap) to beautify the homepage.
To realize realtime search suggestion, we built a new solr core, which includes the papers of lots of citations and is much smaller than the former one, because if we use the former core, it will take too much time to search. And we need to make a search operation when the user press a key on the keyboard, if it is too slow, the experience of the user will be stuck. The solr core returns a json array to the homepage, and the homepage will show it in a droplist.

```
server {

        server_name acemap.lifanz.cn;

        location / {
                root /root/acemap;
                index index.html index.htm index.php;
                try_files $uri $uri/ =404;
        }
        location ~ \.php$ {
 #               include snippets/fastcgi-php.conf;
                try_files       $uri = 404;
                root    /root/acemap;
                fastcgi_pass 127.0.0.1:9001;
#               fastcgi_pass unix:/run/php/php7.2-fpm.sock;
                fastcgi_index index.php;
                fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name$fastcgi_script_name;
                include fastcgi_params;
        }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/lifanz.cn/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/lifanz.cn/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}


server {
    if ($host = acemap.lifanz.cn) {
        return 301 https://$host$request_uri;
    } # managed by Certbot



        server_name acemap.lifanz.cn;

        listen 80;
    return 404; # managed by Certbot


}
```

Figure 2: The configuration of Nginx

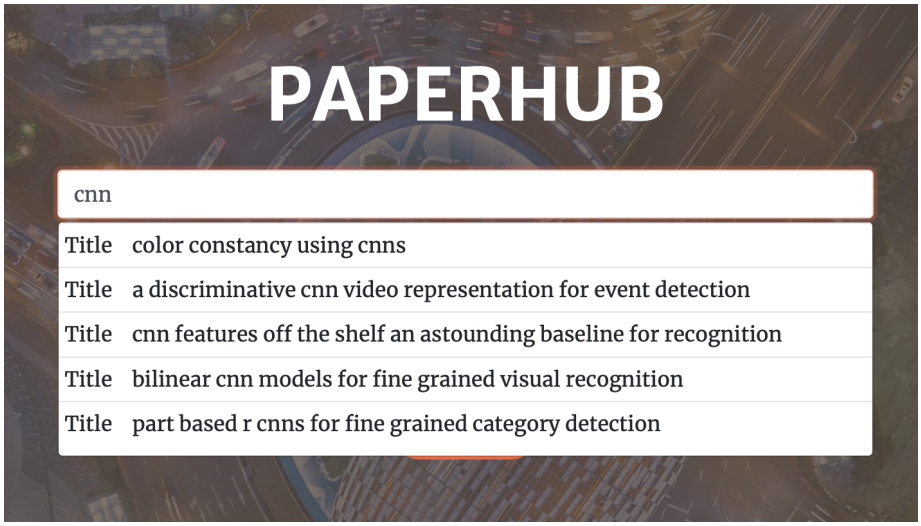


Figure 3: The search suggestion droplist

3 About Search Page

3.1 The Page Layout

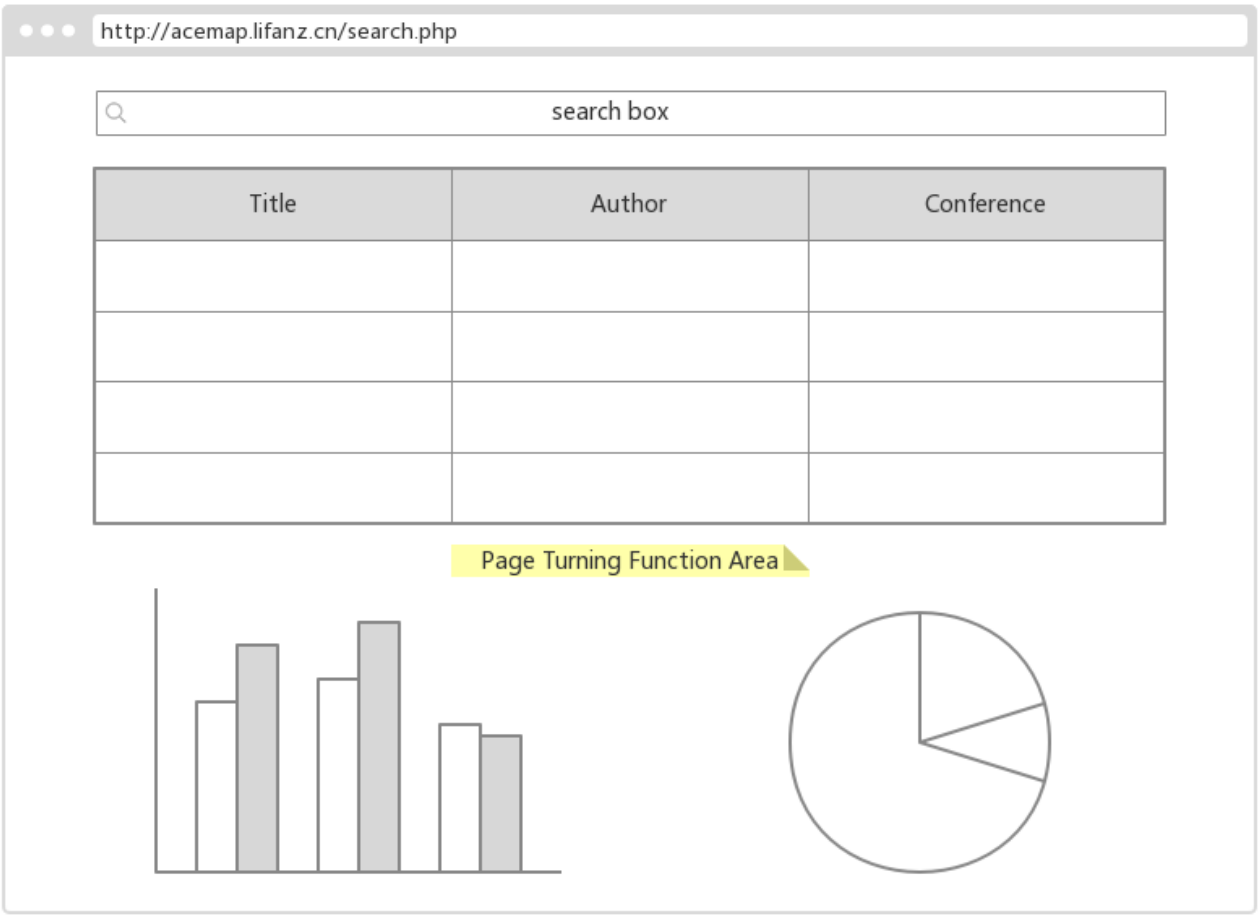


Figure 4: layout

The layout of this page is shown in Figure 4. In the top area, we set a search box. In the middle of the page, we use a table to show the search result. And the function area (to turn the page) is just below the table. At the bottom of the page, we show two statistical figures (according to the search result).

How to realize this layout?
Use bootstrap grid layout. Shown in figure 5.

```
<div class="col-md-12 col-xs-12 col-sm-12 panel panel-default" id="table_div">
<div class="col-md-6 col-xs-6 col-sm-6 panel panel-default" id="image_div" style="background-color:rgba(255,255,255,0.9);height:500px;border:0px;">
```

Figure 5: code

3.2 Using Ajax To Realize Interaction

Why we choose ajax?

- Reload the changing part instead of reloading the whole page ⇒ more fluent response
- Separate the front-end and the back-end ⇒ more clear web design logic

How the ajax work in our search page?
Shown in figure 6.

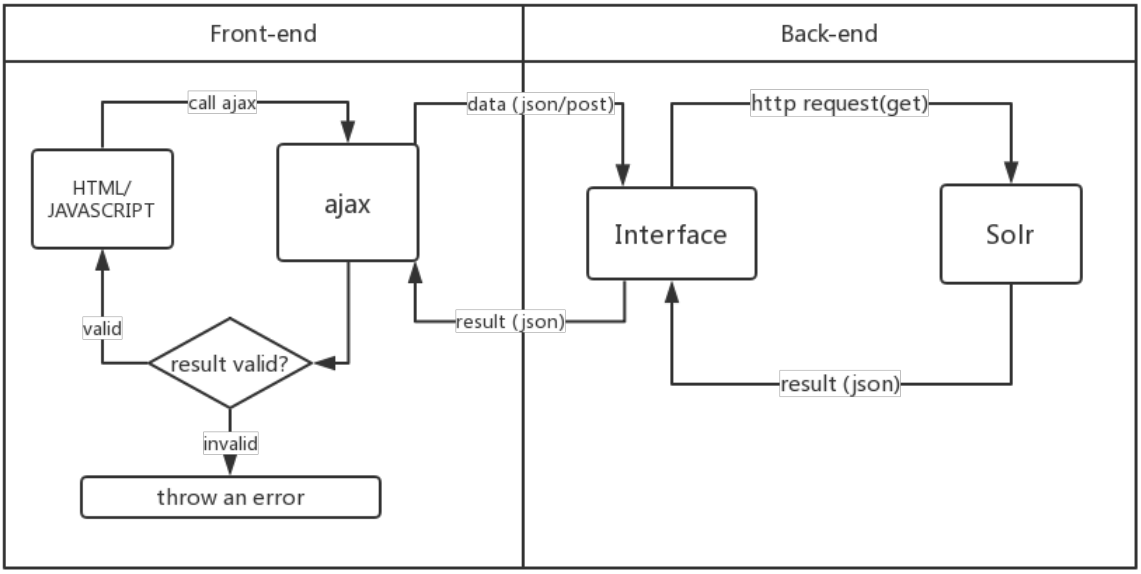


Figure 6: ajax

How to realize it?
JQuery.ajax: Just like the code shown in figure 7

```
$.ajax({
    type: "POST",
    async: "true",
    url: "./search_solr.php",
    dataType: "json",
    data: {
        "field":global_field,
        "value":global_value,
        "start":global_start,
        "rows":global_rows,
    },
    success: function(msg) {
        to_show(msg);
    },
    error: function(XMLHttpRequest, textStatus, errorThrown) {
        console.log("Error!" + " " + XMLHttpRequest.status + " " + XMLHttpRequest.readyState + " " + textStatus);
    }
});
```

Figure 7: ajax code

Interface: search_solr.php: Just like the code shown in figure 8

```php
<?php
    $field = $_POST["field"];
    $value = $ POST["value"];
    if(array_key_exists("start", $_POST)){
        $start = $_POST["start"];
    }else {
        $start = 0;
    }
    if(array_key_exists("rows", $_POST)){
        $rows = $_POST["rows"];
    }else {
        $rows = 10;
    }
    $str = urlencode(str_replace(' ','+',$field.":".$value));
    $ch = curl_init();
    $timeout = 10;
    $url = "http://acemap.lifanz.cn:8983/solr/ee101_core_1/select?indent=on&q=".$str."&rows=".$rows."&start=".$start."&wt=json";
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
    $result=json_decode(curl_exec($ch), true);
    curl_close($ch);
    //echo $url;
    echo json_encode($result);
?>
```

Figure 8: search_solr.php code

Color	Function
green	receive data (by post)
yellow	set default value
blue	search in solr (by http request)
red	return the result (in json format)

3.3 Paging

3.3.1 Model 1

UI Design

Figure 9: paging ui design

Imitate google and baidu (figure 9). Associated code: (figure 10)

```javascript
function turn_page_button(msg){
    var str = "";
    str += "Found " + msg["response"]["numFound"] + " results<br/>";
    var this_page = parseInt(global_start/global_rows) + 1;
    var max_page = parseInt((parseInt(msg["response"]["numFound"])-1)/global_rows) + 1;
    var lower_bound = this_page - 5;
    if(lower_bound <= 0){lower_bound = 1;}
    var upper_bound = lower_bound + 9;
    while(upper_bound * global_rows >= msg["response"]["numFound"] + global_rows){upper_bound--;}
    str += "<ul class=\"pagination center-block\">";
    str += "<li><a href=\"javascript:turn_page(1)\">&lt;&lt;</a></li>";
    for(var i = lower_bound; i <= upper_bound; i++){
        if(i == this_page){
            str += "<li class=\"active\"><a href=\"javascript:turn_page("+String(i)+")\">"+String(i)+"</a></li>";
        }else{
            str += "<li><a href=\"javascript:turn_page("+String(i)+")\">"+String(i)+"</a></li>";
        }
    }
    str += "<li><a href=\"javascript:turn_page("+max_page+")\">&gt;&gt;</a></li></ul>";
    return str;
}
```

Figure 10: paging ui code

How to realize the paging function?

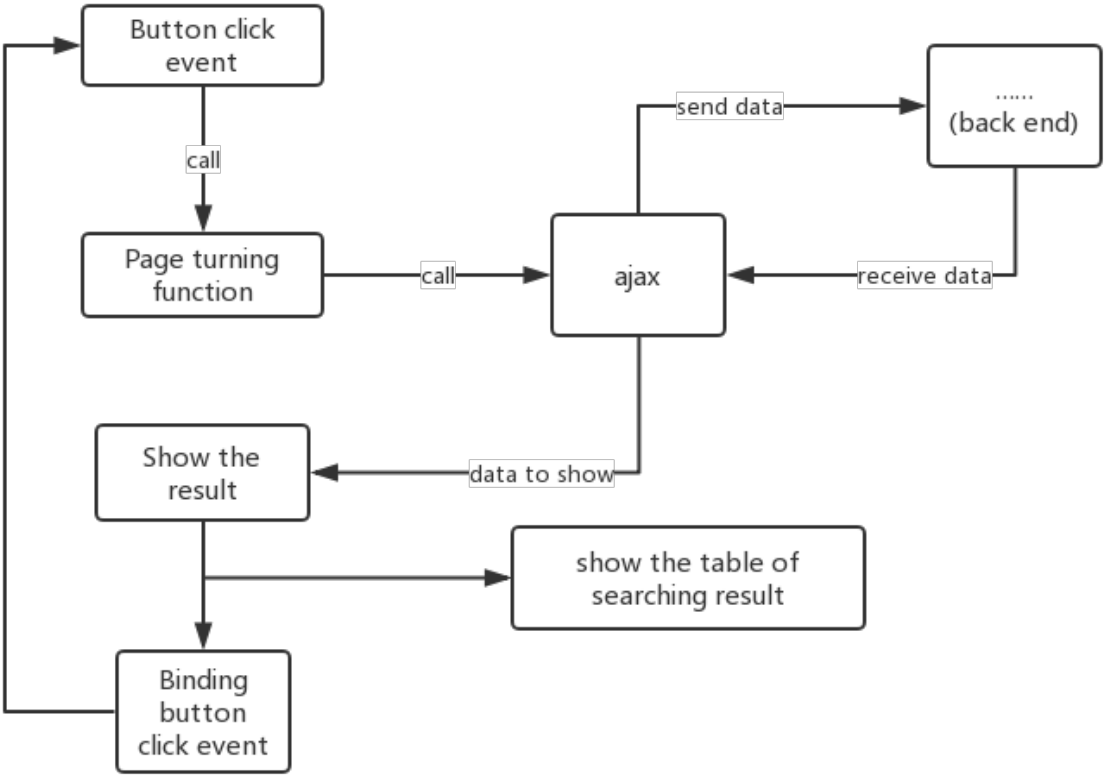


Figure 11: page turning

How to reload the table when turning page?
We use a simple way: $document.getElementById('div\_id').innerHTML = str$; where 'str' is a string variable which contains the html codes to show in the new table area. And apparently, the string variable 'str' is produced by the program dynamically according to the search result returned by solr.

Model 2

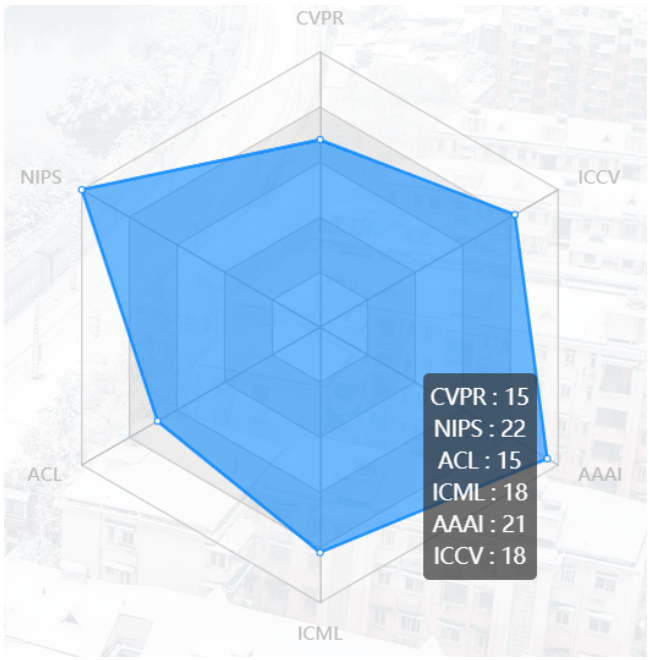
Move your mouse to the right side of the searching result table and you will see your cursor changing into an arrow, then if you click your mouse, it will turn to the next page. Similarly if you move your mouse to the left side and click it, the web page will turn to the last page.

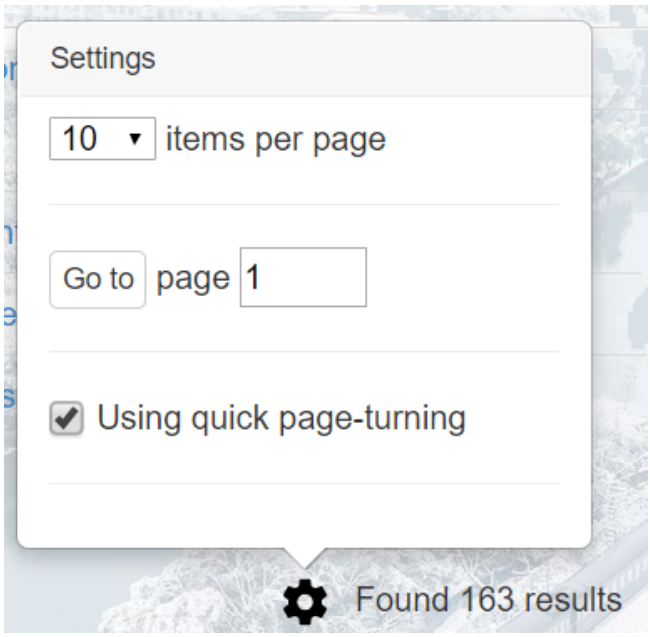
We realize it in this way: Monitor the mouse moving event. If the cursor is in the detecting area, we change the cursor style in an arrow(left arrow or right arrow) and binding a turning page function to the mouse click event. When your cursor leave this detecting area, we unbind the click event.

3.4 Highlight

Search box	Set a search box at the top of this page. Users can change the searching words in a more convenient way.
Radar map	There are 13 conferences. And the radar map only show the top 6 conferences. So we need to sort them. But after sorting the conferences the shape of the radar map graph will be monotonous. Randomization is the solution.

Settings	Using bootstrap popover. Users can click the setting button, then set the number of rows to show, go to a certain page or turn on/off the second paging model.
User-friendly reloading	Using ajax to realize paging, there is a problem: when the web page reload, it will go to the first page. But we hope it can remain its original page instead of the first page. To solve this problem, we use session. When the 'onbeforereload' event triggered we write the page records into the session. When 'onload' event triggered we read the records from session.(We have also tried cookie, but session is much better.)





4 About Using Echarts (Author Page And Affiliation Page)

How can we draw statistical graph using 'Echarts'?

- Know the 'Echarts' API;
- Processing and formatting data;
- Setting the tooltip;
- Adjust the layout.

Know the Echarts' API:

Such as echarts.init(), echarts.setOption(), echarts.legend, echarts.tooltip, echarts.event. We can get the direction for use in the official API document.

Processing and formatting data:

After we decide in which type of diagram to show the data we get, we should format the data to make it fit the diagram. Different types of diagram require different data format. We can know the required data format for a certain diagram type in the way shown below(figure 12)

We write the code 'console.log(data);' in the left coding area. And we can find the output in the browser console. And we can see in this case the data is formatted in JSON, and it expresses the tree relationship by recursively nested arrays.

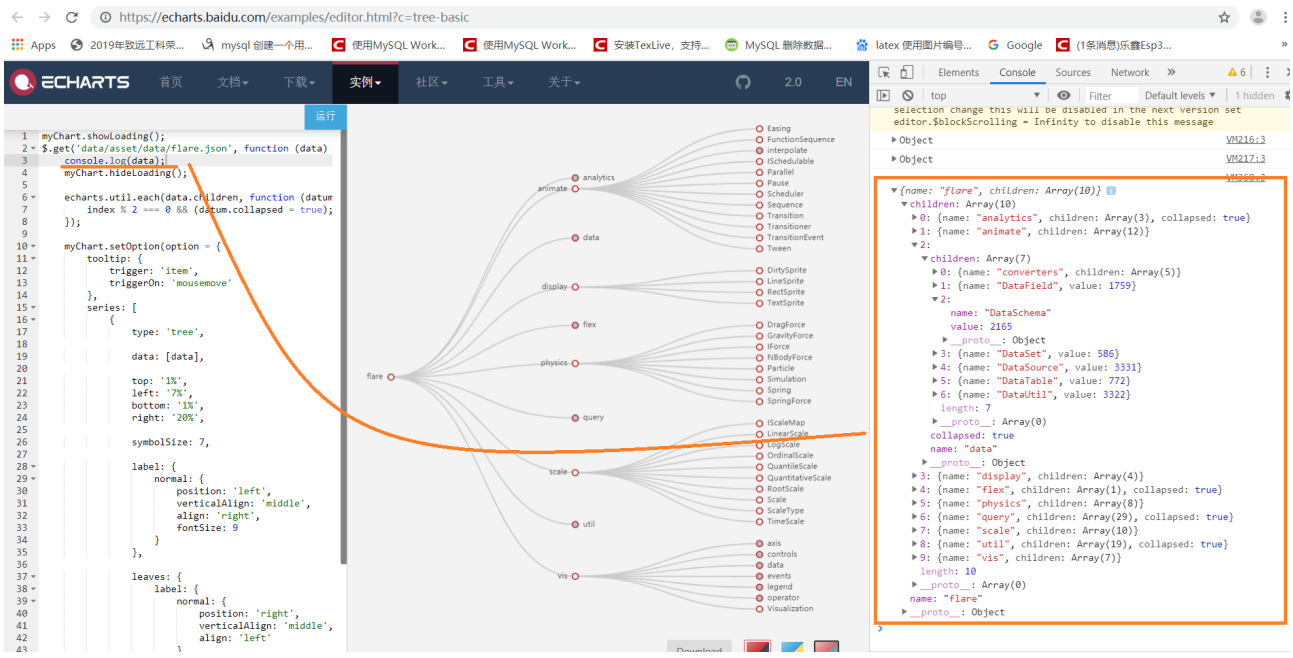


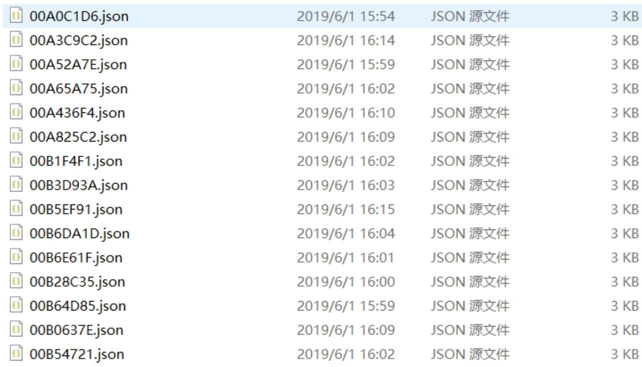
Figure 12: Know the required data format

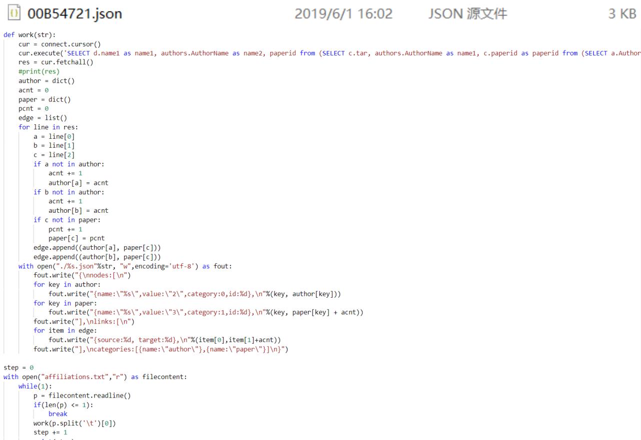
```php
<?php
    $author_id = $_POST["author_id"];
    $author_name = $_POST["author_name"];
    $retval = array();
    $retval["name"] = ucwords($author_name);
    $retval["children"] = array();
    $link = mysqli_connect("db.lifanz.cn:3306", 'ee101_user', 'ee1012019', 'ee101');
    $result = mysqli_query($link, "SELECT g.PaperID as paperid, g.AuthorName as authorname, g.AuthorSequence as seq from ((SELECT d.PaperID , e.AuthorName , d.AuthorSeque
    $tmp_p = "";
    $paper_node = array();
    $idx = 1;
    while($row = mysqli_fetch_array($result)){
        if($tmp_p != $row["paperid"]){
            if($tmp_p != ""){
                $retval["children"][] = $paper_node;
            }
            $paper_node = array();
            $paper_node["name"] = "No.".$idx;
            $idx += 1;
            $paper_node["children"] = array();
        }
        $tmp_p = $row["paperid"];
        if($row["authorname"] != $author_name){
            $authornode = array("name" => "No.".$row["seq"]." ".ucwords($row["authorname"]), "children" => array());
            $paper_node["children"][] = $authornode;
        }
    }
    if($tmp_p != ""){
        $retval["children"][] = $paper_node;
    }
    echo json_encode($retval);
?>
```

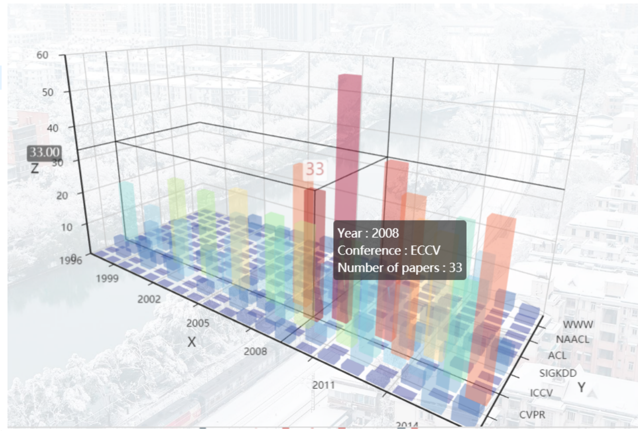
Figure 13: source code

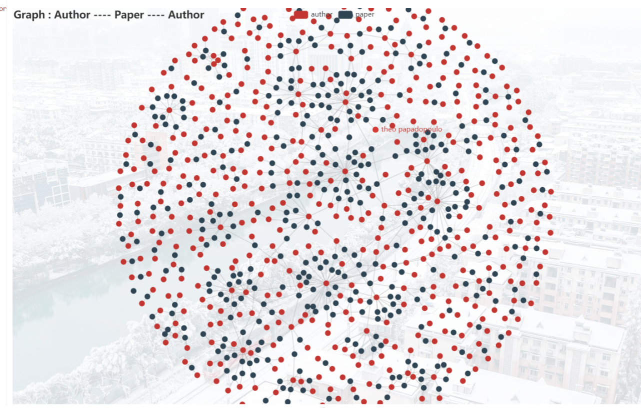
We process the data in a specialized php program. To process a tree graph, I wrote the code below.(figure 13)

We search the data in solr or in mySQL, and deal with data by php or javascript, then show the diagram. When the data is relatively simple, it works well. But sometimes the data is too complex to deal with in this way. Maybe the scale of data is too large, or maybe the data is gotten from the relatively complex calculating. In these cases, if we deal with the data dynamicly, the page will be very slow and not smooth. So for some complex graph (such as 3D bar graph and force oriented author map), we use this method: preprocess the data off-line, but render the graph dynamicly. We preprocess the data by c++ (reading data information from the txt file) and python (getting data information from the mySQL).


00A0C1D6.json 2019/6/1 15:54 JSON 源文件 3 KB
00A3C9C2.json 2019/6/1 16:14 JSON 源文件 3 KB
00A52A7E.json 2019/6/1 15:59 JSON 源文件 3 KB
00A65A75.json 2019/6/1 16:02 JSON 源文件 3 KB
00A436F4.json 2019/6/1 16:10 JSON 源文件 3 KB
00A825C2.json 2019/6/1 16:09 JSON 源文件 3 KB
00B1F4F1.json 2019/6/1 16:02 JSON 源文件 3 KB
00B3D93A.json 2019/6/1 16:03 JSON 源文件 3 KB
00B5EF91.json 2019/6/1 16:15 JSON 源文件 3 KB
00B6DA1D.json 2019/6/1 16:04 JSON 源文件 3 KB
00B6E61F.json 2019/6/1 16:01 JSON 源文件 3 KB
00B28C35.json 2019/6/1 16:00 JSON 源文件 3 KB
00B64D85.json 2019/6/1 15:59 JSON 源文件 3 KB
00B0637E.json 2019/6/1 16:09 JSON 源文件 3 KB
00B54721.json 2019/6/1 16:02 JSON 源文件 3 KB




Year : 2008
Conference : ECCV
Number of papers : 33
Z
X
Y
WWW
NAACL
ACL
SIGKDD
ICCV
CVPR


Graph : Author ---- Paper ---- Author

5 Paper Page

The paper page can be visited through the link given in the *Search Page* and *Author Page* by pass the value of *paper_id* to *Paper.php*. The *Paper Page* consists of three parts : *Basic Information Part*, *Reference Part* and *Citation Part*. The Page is built on *bootstrap*, several *div* and *rows* are added into a *div* whose class is *container* to make the Page tidy and clear.

5.1 Basic Information Part

In this part, the basic information of the selected paper will be listed, including title, authors, published year and conference.

To get all the required information, first, we use this command to get the $ConferenceID$, *title* and *publishyear* of the selected paper.

```
$Base_info=mysqli_fetch_array(mysqli_query($link,"SElECT * from papers where paperid='$Paper_id'"));
```

Then we can get the conference name and all the authors of the paper with the following two SQL command and output them with a hyperlink by which can visit the corresponding $ConferencePage$ and *Paper Page*.

```
echo "<td>";
$au_info=mysqli_query($link,"SElECT AuthorName,AuthorID FROM authors where authorid in (SElECT authorid from
    paper_author_affiliation where PaperId='$Paper_id' order by 'authorsequence')");
while ($rows=mysqli_fetch_array($au_info))
{
    $paperauthor_name=$rows['AuthorName'];
    $paperauthor_id=$rows['AuthorID'];
    echo "<a href=\"/final/author.php?authorid=$paperauthor_id\">$paperauthor_name; </a>";
}
echo "</td>";

echo "<td>";
echo $Base_info['PaperPublishYear'];
echo "</td>";

echo "<td>";
$conf_id=$Base_info['ConferenceID'];
$Conference_info=mysqli_fetch_array(mysqli_query($link,"SElECT ConferenceName from conferences where ConferenceID='$conf_id
    '"));
$conf_name=$Conference_info['ConferenceName'];
echo "<a href=\"/final/conference.php?conferenceid=$conf_id\">$conf_name </a>";
echo "</td>";
```

And here is the basic information table.

Paper

Title	Authors	Year	Conference
the 3d line motion matrix and alignment of line reconstructions	adrien bartoli; peter sturm;	2004	CVPR

Figure 14: Basic Information

5.2 Reference Part

The *Reference Part* includes two parts. The first part is a table showing the detailed information of all the papers which were referred by the selected paper. And the second part is statistics of the reference information and their visualization.

5.2.1 Part I

In *Part I*, considering that a paper can't refer to too many other papers, so we choose to list all the reference information in form of table. All the information is ordered by the publish year of the referred papers. In the table $Conference$, *Title* and *Author* of the referred paper are given with links so that people can visit the corresponding page by clicking the link.

Reference Information

Title	Authors	Year	Conference
real time object detection for smart vehicles	vasanth philomin; dariu m gavrila;	1999	ICCV
efficient matching of pictorial structures	pedro f felzenszwalb; daniel p huttenlocher;	2000	CVPR
learning to parse pictures of people	remi ronfard; bill triggs; cordelia schmid;	2002	ECCV
detecting pedestrians using patterns of motion and appearance	jones; viola; snow;	2003	ICCV
pca sift a more distinctive representation for local image descriptors	rahul sukthankar; yan ke;	2004	CVPR
human detection based on a probabilistic assembly of robust part detectors	andrew zisserman; krystian mikolajczyk; cordelia schmid;	2004	ECCV
a performance evaluation of local descriptors	krystian mikolajczyk; cordelia schmid;	2005	CVPR

Figure 15: Table of Information

The process of getting the required information is quite similar to the one in the $BacisInformationPart$. First, we use the following command to get the $PaperID$, $ConferenceName$, $ConferenceID$, $Title$ and $PublishYear$ of the referred papers.

```
$ALL_Refer_info=mysqli_query($link,"SELECT papers.paperid as PaperID, papers.title as Title, conferences.ConferenceID
    as ConferenceID, conferences.ConferenceName as ConferenceName, papers.PaperPublishYear as Year from papers inner
    join conferences inner join paper_reference on papers.paperid=paper_reference.referenceid and papers.ConferenceID=
    conferences.ConferenceID and paper_reference.paperid='$Paper_id' order by papers.PaperPublishYear");
```

Figure 16: get referred papers

```
while($rows=mysqli_fetch_array($ALL_Refer_info))
{

    echo "<tr>";

    echo "<td>";
    $refer_title=$rows['Title'];
    $refer_id=$rows['PaperID'];
    echo "<a href=\"/Paper.php?paper_id=$refer_id\">$refer_title</a>";
    echo "</td>";

    echo "<td>";
    $ref_auinfo=mysqli_query($link,"SELECT AuthorName,AuthorID FROM authors where authorid in (SElECT authorid from
        paper_author_affiliation where PaperId='$refer_id' order by 'authorsequence')");
    while($row2=mysqli_fetch_array($ref_auinfo))
    {
        $refer_authorname=$row2['AuthorName'];
        $refer_authorid=$row2['AuthorID'];
        echo "<a href=\"/author.php?authorid=$refer_authorid\">$refer_authorname; </a>";
    }
    echo "</td>";

    echo "<td>";
    echo $rows['Year'];
    echo "</td>";

    echo "<td>";
    $refer_confid=$rows['ConferenceID'];
    $refer_confname=$rows['ConferenceName'];
    echo "<a href=\"/conference.php?conferenceid=$refer_confid\">$refer_confname </a>";
    echo "</td>";

    echo "</tr>";
}
```

Figure 17: output the result with hyperlink

Then Similarly, for each referred paper, we use another SQL command to get the authors from the table $paper\_author\_affiliation$ and table $authors$. Then we represent them in form of table with hyperlink which links to the corresponding pages.

5.2.2 Part II

In $Part\ II$, three charts are given, including a pie chart, a histogram and a Reference tree, which represent the statistics of different data.

Pie Chart

The first part shows the distribution of the papers in different conferences. Put the mouse on the figure and it will show the detailed number.

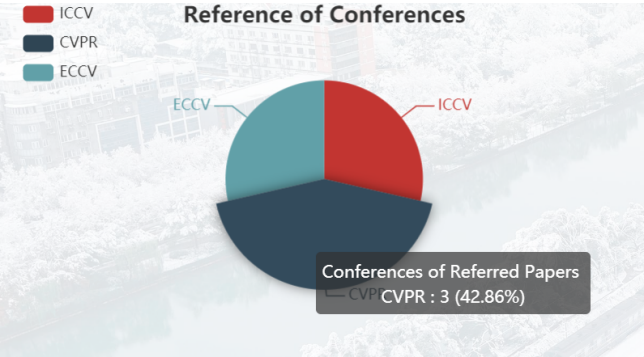


Figure 18: Pie figure

To get the required Information, we use *ajax* to connect another *php* and post the necessary parameter to it. And the *ajax* will run the function *refer_bing_show()*, which use *Echarts* to visualize the data in form of pie chart.

```
$.ajax(
        {
            type:"POST",
            async:"false",
            url:"./get_refer_confer.php",
            data:{"paper_id":"<?php echo $Paper_id; ?>"},
            dataType:"json",
            success:function(msg){refer_bing_show(msg);},
            error: function(XMLHttpRequest, textStatus, errorThrown)
            {
                alert("Error!" + XMLHttpRequest.status + XMLHttpRequest.readyState + textStatus);
            }
        }
    )
```

In *get_refer_confer.php*, we use following *SQL* commands to get the names of the conferences and the referred papers. Then we make a simple calculation of the distribution of papers in different conference. Then we return the result in *json* form.

```
$Refer=mysqli_query($link,"SELECT conferences.ConferenceName as ConferenceName,conferences.conferenceid as ConferenceID, papers.Title as Title
    from conferences inner join paper_reference inner join papers on paper_reference.paperid='$Paper_id' and paper_reference.ReferenceID=papers.
    paperid and papers.conferenceid=conferences.conferenceid");
$conf_sum=array();
$cf_id=array();
while($rows=mysqli_fetch_array($Refer))
{
    $cf_id[$rows['ConferenceName']]=$rows['ConferenceID'];
    if(array_key_exists($rows['ConferenceName'], $conf_sum))
        $conf_sum[$rows['ConferenceName']]+=1;
    else $conf_sum[$rows['ConferenceName']]=1;
}
```

Figure 19: *SQL* commands

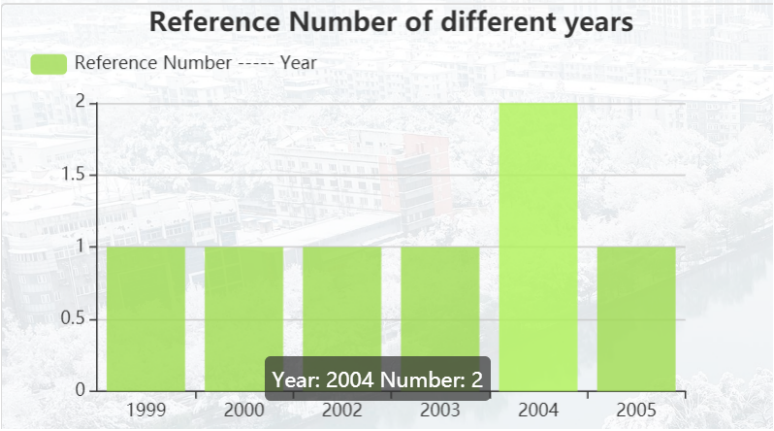
```
myCharts_b1.on('click', function (param){window.location.href="/conference.php?conferenceid="+param.data.id;});
```

Figure 20: function *on*()

Also, the function *on*() of *Echarts* is redefined, which adds a hyperlink to the corresponding part in the chart so that people can visit *ConferencePage* by clicking the sector.

Histogram

The second figure shows how many papers are referred in different years. Put the mouse on the figure and we can see the detailed information. The selected part is highlighted.



Similarly, we use the *ajax* to get necessary information.And in *get_refer_year.php*, we use this *SQL* command to get all the referred papers' publish year and use several loop statements to make a statics. Then we return the result to *ajax* in form of *json*.

```
$.ajax(
    {
        type:"POST",
        async:'false',
        url:'./get_refer_year.php',
        data:{'paper_id':"<?php echo $Paper_id;?>"},
        dataType:'json',
        success:function(msg){refer_zhu_show(msg);},
        error: function(XMLHttpRequest, textStatus, errorThrown)
        {
            alert("Error!" + XMLHttpRequest.status + XMLHttpRequest.readyState + textStatus);
        }
    }
)
```

(a) *ajax*

```
$year_sum=array();
while($rows=mysqli_fetch_array($Refer_Y))
{
    if(array_key_exists($rows['Year'], $year_sum))
        $year_sum[$rows['Year']]+=1;
    else $year_sum[$rows['Year']]=1;
}
$data_y=array();
$data_n=array();
foreach ($year_sum as $key => $value)
{
    $data_n[]=$value;
    $data_y[]=$key;
}
echo json_encode(array($data_y,$data_n));
```

(b) Statics

```
$Refer_Y=mysqli_query($link,"SELECT papers.paperpublishyear as Year, papers.Title as Title from paper_reference inner join papers on
    paper_reference.paperid='$Paper_id' and papers.paperid=paper_reference.referenceid order by papers.paperpublishyear");
```

Figure 21: *SQL* command

Also, we the *toolip* in the option of *Echarts* is redesigned as follows:

```
tooltip:
{
    formatter: function(param, idx)
    {
        return "Year: " + data[0][param["dataIndex"]] + "\nNumber: " + param["data"];
    }
},
```

Then once we put our mouse on the chart, we can see the detailed number of each year.

Reference Tree

The third part is the reference tree, which shows the reference relations clearly. The process of getting data is quite similar with the previous two. A *ajax* is also used in this part. In *get_refer.php*, we select the titles and the conferences' names of the referred papers by this *SQL* command. Then we sort then by their conferences' names and gather then in the layout of a tree.

```
$Refer=mysqli_query($link,"SELECT papers.PaperPublishYear as Year, papers.Title as Title, conferences.ConferenceName as ConferenceName from
    papers inner join conferences inner join paper_reference on papers.PaperID=paper_reference.ReferenceID and papers.ConferenceID=conferences.
    ConferenceID and paper_reference.PaperID='$Paper_id'");
```

Figure 22: *SQL* command

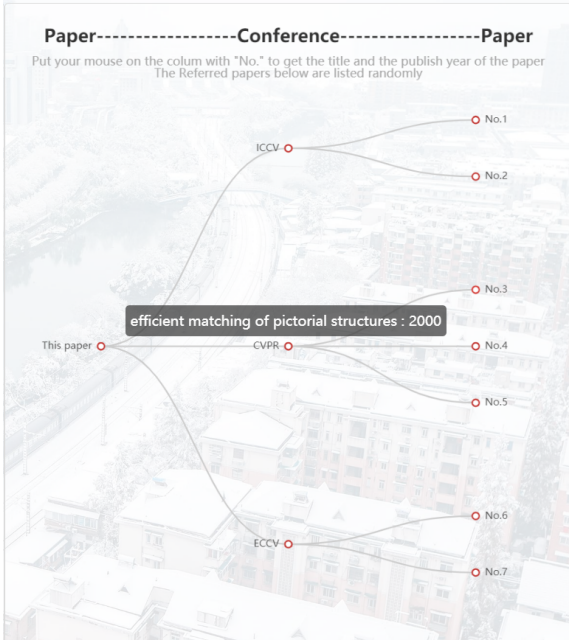
```php
$refer_rela=array();
while($rows=mysqli_fetch_array($Refer))
{
    if(array_key_exists($rows['ConferenceName'],$refer_rela))
        $refer_rela[$rows['ConferenceName']][]=$rows['Title']." : ".$rows['Year'];
    else
    {
        $refer_rela[$rows['ConferenceName']]=array();
        $refer_rela[$rows['ConferenceName']][]=$rows['Title']." : ".$rows['Year'];
    }
}
$Refer_node=array();
$Refer_node['name']='This paper';
$Refer_node['children']=array();
$idx=1;
$confs=array();
foreach ($refer_rela as $conf_n => $paper_n)
{
    $conf_node=array();
    foreach ($paper_n as $tit)
    {
        $conf_node[]=array('name' => 'No.'.$idx, 'children'=> array());
        $confs[]=$tit;$idx+=1;
    }
    $Refer_node['children'][]=array("name" => $conf_n, "children" => $conf_node);
}
```

Figure 23: Sorting

The titles of the papers are very long, which means it's really inconvenient to print them on the screen directly. So we redesigned the *toolip* in the option of the chart. We use *No.* with number to represent the papers on the reference tree and The number of each paper is given randomly. When the visitors of the Page put their mouse on the node of papers, the detailed information of paper, including title and the publish year of papers will be shown.

```javascript
tooltip:
{
    trigger: 'item',
    triggerOn: 'mousemove',
    formatter: function(param)
    {
        var tmp=param['data']['name'];
        var lis=tmp.split('.');
        if(lis.length>1)
            return data[0][parseInt(lis[1])-1];
        else return tmp;
    }
},
```

(a) *toolip*



(b) Reference Tree

And because the height of the tree is not fixed so the height of this *div* in *html* is also flexible. The height of this *div* has the relation below.

```php
<div class="col-md-12 col-xs-12 col-sm-12 panel panel-default" style="margin-top:15px;background-color:rgba(255,255,255,0.75);height:
    <?php echo $refer_num * 35+480; ?>px;" >
```

Figure 24: Relation

The more papers are referred, the larger the *div* is so that the charts will not be out of range.

5.3 Citation Part

The *Citation Part* also includes two part. The first part is the citation information in form of tables. And the second part is statistics of the citation information and their visualization. The structure of *Citation Part* is quite similar with the *Reference Part* so we will not show the repeated or similar codes in this part of report. What we will do is to show the different part and the additional function.

5.3.1 Part I

In *Part I*, we just list the newest 15 (at most) citation records in the table, because a paper might be referred by many papers for many times and if we list all information the size of the table will be extremely huge, which makes the Page ugly. The process of getting data and output is similar with the one in *Reference Part*, and just some slight code adjustment can make it. In this part you can see the *Titles*, *Authors*, *Published Years* and *Conferences* of each paper. And to get all the detailed information by clicking the hyperlink.

Figure 25: The hyperlink to the *Detailed Information Page*

The hyperlink given in the figure above gives visitors a way to visit the *detailed information page*.

Detailed Information Page

The *Detailed Information Page* will list all the detailed citation information in form of tables, including *Titles*, *Authors*, *Published Years* and *Conferences*. Considering the size of the page and the data, only 15 lines of information will be listed in one page and the turning page function is designed.

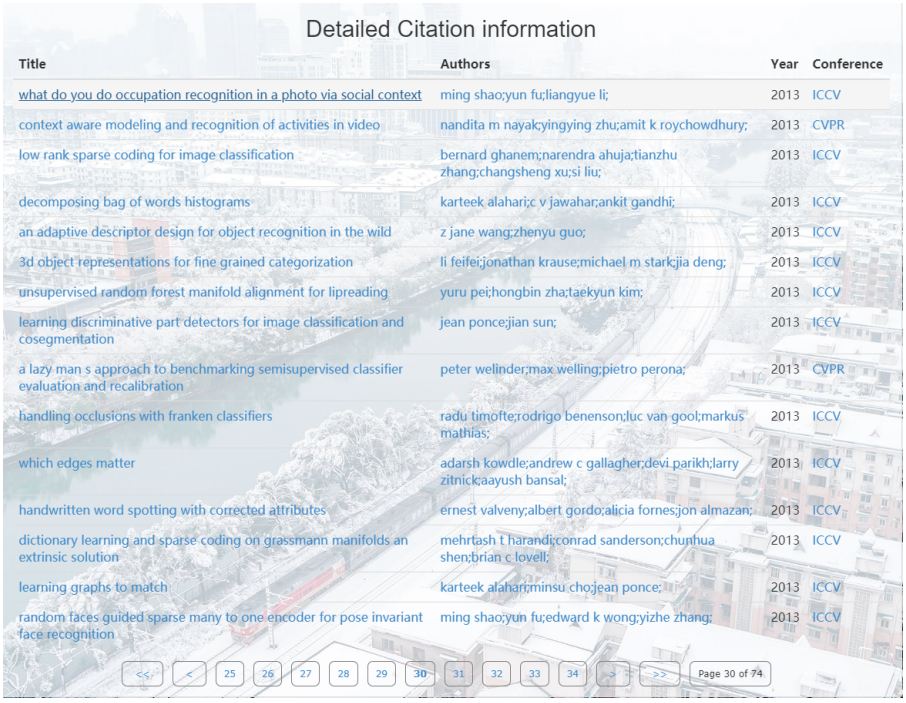


Figure 26: Detailed Information Page

The information in every page is different depending page number, which gives a limited range to the result. So we can't use fixed commands to get necessary information. To do this we must use *function* in our *php* commands.

First, we need to get the *Paper_id* and the *PageNumber* of the current page. They will be posted by the hyperlink. And other basic variables to limit the range are also initialized at the beginning of the program. The @ in the code means ignore the potential warning so that the code can run fluently.

```php
$Paper_id=$_GET['paper_id'];
@$pagesize=15;
@$pagenum=empty($_GET['pagenum'])?1:$_GET['pagenum'];
@$result=get_res($pagenum,$pagesize);
@$pageend=ceil(get_all()/$pagesize);
@$info=$result['res'];
```

Figure 27: Initialization

```php
function get_all()
{
    $Paper_id=$_GET['paper_id'];
    $link=mysqli_connect('db.lifanz.cn:3306','[redacted]','[redacted]','[redacted]');
    $res=mysqli_query($link,"SELECT count(*) as cnt from papers inner join conferences inner join paper_reference on paper_reference.
        referenceid='$Paper_id' and paper_reference.paperid=papers.paperid and papers.ConferenceID=conferences.ConferenceID order by
        papers.PaperPublishYear desc");
    return mysqli_fetch_array($res)['cnt'];
}
```

Figure 28: *get_all*() function

The *get_all*() use *SQL* commands like *SELECT* and *count*(∗) to get the size of data and return it as the result. With the size of the data, we calculate the total pages of the detailed information.

The data to be represent is not fixed depending on the page number. To collect the data, we need another function.

```php
function get_res( $pagenum=1, $pagesize=15)
{
    $array=array();
    $Paper_id=$_GET['paper_id'];
    $link=mysqli_connect('db.lifanz.cn:3306','[redacted]','[redacted]','[redacted]');
    $res=mysqli_query($link,"SELECT papers.paperid as PaperID, papers.title as Title, conferences.ConferenceName as ConferenceName,
        conferences.ConferenceID as ConferenceID, papers.PaperPublishYear as Year from papers inner join conferences inner join
        paper_reference on paper_reference.referenceid='$Paper_id' and paper_reference.paperid=papers.paperid and papers.ConferenceID=
        conferences.ConferenceID order by papers.PaperPublishYear desc limit ".(($pagenum-1)*$pagesize).','.$pagesize);
    $idx=0;
    while($rows=mysqli_fetch_array($res))
    {
        $array[$idx]=array('paperid'=>$rows['PaperID'],'conferenceid'=>$rows['ConferenceID'],'title'=>$rows['Title'],'Year'=>$rows['
            Year'],'confn'=>$rows['ConferenceName'],'author'=>array());
        $cita_id=$rows['PaperID'];
        $au_info=mysqli_query($link,"SELECT AuthorName,AuthorID FROM authors where authorid in (SElECT authorid from
            paper_author_affiliation where PaperId='$cita_id' order by 'authorsequence')");
        while($row2=mysqli_fetch_array($au_info))
            $array[$idx]['author'][]=array('name'=>$row2['AuthorName'],'id'=>$row2['AuthorID']);
        $idx+=1;
    }
    mysqli_close($link);
    return array('res'=>$array);
}
```

Figure 29: *get_res*() function

The *get_res*() function use *SELECT* command with parameter *limit* to get the data to be represented. In *SQL* commands, *Limit* is used in form of $\{Limit\ a, b\}$, which means this *SQL* will select from the $No.a$ result to the $No.(a+b-1)$ result and return them as the final result of the query (The number of the results starts from 0, which means $a \geq 0$). The result of the *SQl* query includes $PaperID$, $ConferenceID$, $Title$, $ConferenceName$ and $PublishedYear$. The $PaperID$ and $ConferenceID$ is used for adding hyperlink. And to get the authors of the papers, we use another *SQL* command in the loop the get information from the table *Authors* and *paper_author_affiliation*. Then we merge them all together and return it as the result of the function *get_res*(). The output process is almost the same as the one in *Part I* of $Reference\ Part$. Only some small adjustment in codes can make it, so we will not do any more introduction about it.

At the end of this page is the buttons for turning pages. Every button has hyperlink goes to the corresponding page. The button with < and > goes to the previous and next one page, the button with << and >> goes to the first and the last page. Also ten buttons is listed, which goes to the 10 pages nearby the current page. The text on the button of the current page is emphasized.

Figure 30: buttons

The style of the buttons is redesigned with *CSS*.

```css
.mypaging
{
    opacity: 1;
    margin:0px 12px 0px 0px;
    padding: 8px 10px;
    font: 12px Verdana;
    border : 1px solid grey;
    border-radius : 7px;
}
```

(a) *CSS*

```php
<?php
    $Paper_id=$_GET['paper_id'];
    $las=($pagenum==1?1:$pagenum-1);$nex=($pagenum==$pageend?$pageend:$pagenum+1);
    echo "<a class = \"mypaging\" style = \"padding : 8px 16px;\" href =\"?paper_id=$Paper_id&pagenum=1\"><<</a>";
    echo "<a class = \"mypaging\" style = \"padding : 8px 16px;\" href =\"?paper_id=$Paper_id&pagenum=$las\"><</a>";
    for($x=($pagenum>5?$pagenum-5:1);$x<min($pagenum+5,$pageend+1);++$x)
        echo "<a class = \"mypaging\" href = \"?paper_id=$Paper_id&pagenum=$x\">".(($x == $pagenum) ? "<strong>" : "").$x.(($x
            == $pagenum) ? "</strong>" : "")."</a> ";
    echo "<a class = \"mypaging\" style = \"padding : 8px 16px;\" href =\"?paper_id=$Paper_id&pagenum=$nex\">></a>";
    echo "<a class = \"mypaging\" style = \"padding : 8px 16px;\" href =\"?paper_id=$Paper_id&pagenum=$pageend\">>></a>";
    echo "<span class = \"mypaging\">Page $pagenum of ".$pageend."</span>";
?>
```

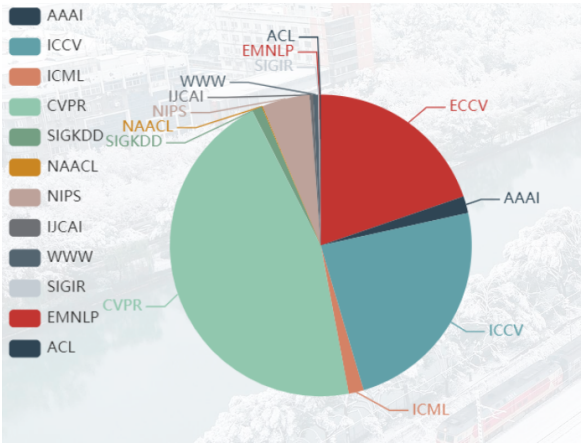
(b) Codes of buttons

5.3.2 Part II

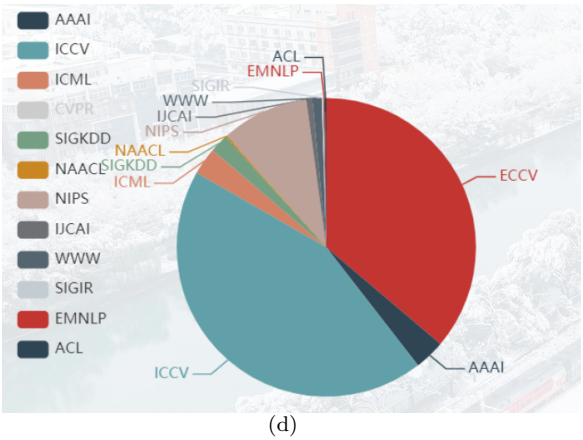
The Second Part of *Citation Part* also consists of three parts: a pie chart, a histogram and a Citation tree, which represents the statics of the data.

Pie Chart

The pie chart shows the distribution of the papers in different conferences. The process of getting necessary information is almost the same as the one in the *Reference Part*. Just some small adjustment of codes can make it. I want to show that the number of the conferences may be large and the proportion of the papers in different conferences might be not so balanced so some sector on the pie chart might be difficult to be seen. In this situation, visitors can click the legends on the left of the chart then the chart will hide the corresponding sector and the legend will turn grey. if visitors want to see the hidden sectors, just click the legend again.



(c)



(d)

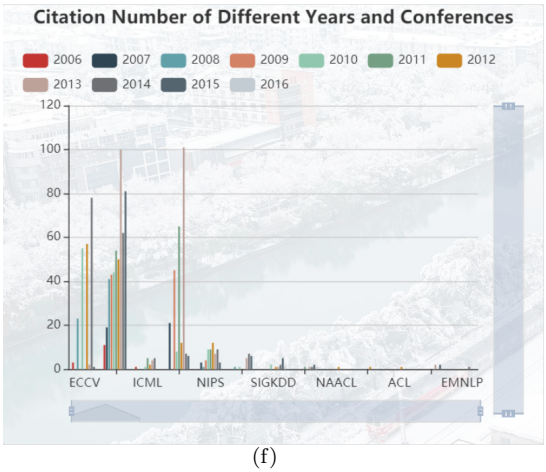
This will make it more convenient to click or read the detailed data.

Histogram

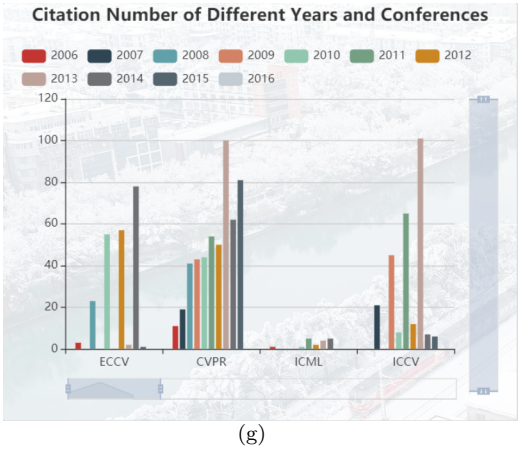
The histogram in *Citation Part* is a little bit different this time we count the citation in different year and different conferences. Put the mouse on the histogram, the year and the exact number of the selected part will be shown. To get the necessary information, we use *ajax* to post the *Paper_id* to a *php* and use *SQL* command to get the conferences' names and publish years of every citation. We find that the distribution of papers might be dispersive, which makes the chart difficult to be seen clearly. so a *datazoom* is added into the option of the chart.

```
dataZoom: [
        {
            show: true,
            start: 0,
            end: 100,
            height:'5%',
            bottom:'5%'
        },
        {
            type: 'inside',
            start: 0,
            end: 100
        },
        {
            show: true,
            yAxisIndex: 0,
            filterMode: 'empty',
            width: 30,
            height: '70%',
            showDataShadow: false,
            left: '93%'
        }
    ],
```

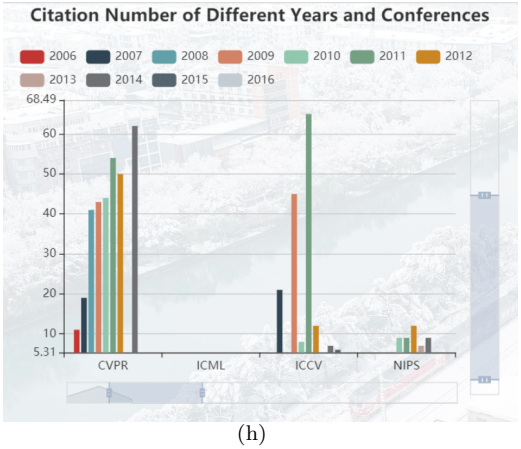
(e) *datazoom*



(f)



(g)



(h)

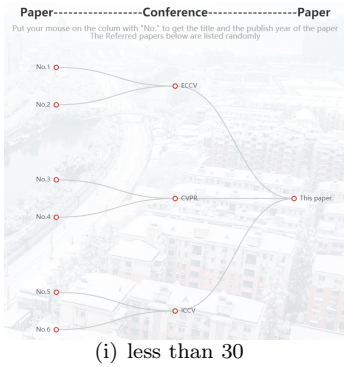
Datazoom give the chart two sliders. visitors can control the size of the chart by the sliders, also they can control it by the mouse wheel. Once the visitors wants to see the part of the whole chart, they can enlarge the chart. But we need to emphasize that the mouse wheel can only control the zooming of the x-axis.

Citation Tree

The citation tree shows the citation relations directly, But we know that the citation tree might be very big, which makes the page impossible to be seen clearly. So we adjust the type of the tree depending on the size of data. When there are more than thirty lines of citation information the tree will be displayed by radial type. To make it, we change the *layout* of *series* in the option of the chart into *radial*.

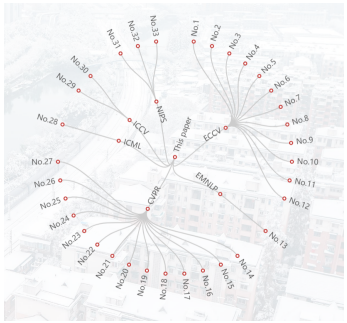
```
series: [
    {
        type: 'tree',
        data: [data[1]],
        top: '60px',
        left: '15%',
        bottom: '15px',
        right: '15%',
        symbolSize: 5,

        layout: 'radial',
```



(i) less than 30

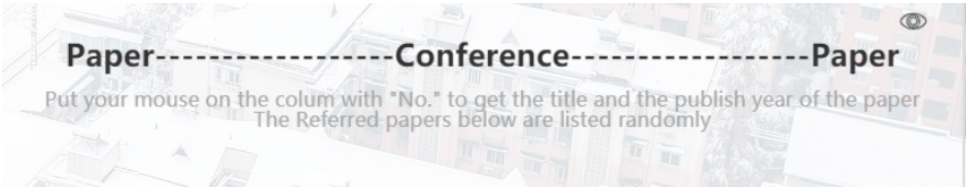
(j) greater than 30(depth=2)



(k) greater than 30(depth=3)

And to avoid the situation that too many papers are represents at the same time, the initial depth of the radial tree is set as 2. The visitors can expand the tree by clicking the nodes on the tree.

For the situation that we need to use radial tree, we design another page for visitors to see the whole tree in normal way. Click The symbol of eye then the visitors can go to the page of big citation tree.



To realize the function, we design one tool in the toolbox of the chart. The eye is drawn by *Path* which use coordinate and other parameter to represent lines and points.

```
toolbox:
{
    feature:
    {
        myTool1:
        {
            show:true,
            title:'click',
            icon: 'path://M432.45,595.444c0,2.177-4.661,6.82-11.305,6.82c-6.475,0-11.306-4.567-11.306-6.82s4.85
            2-6.812,11.306-6.812C427.841,588.632,432.452,593.191,432.45,595.444L432.45,595.444z M421.155,589.87
            6c-3.009,0-5.448,2.495-5.448,5.572s2.439,5.572,5.448,5.572c3.01,0,5.449-2.495,5.449-5.572C426.604,5
            92.371,424.165,589.876,421.155,589.876L421.155,589.876z M421.146,591.891c-1.916,0-3.47,1.589-3.47,3
            .549c0,1.959,1.554,3.548,3.47,3.548s3.469-1.589,3.469-3.548C424.614,593.479,423.062,591.891,421.146
            ,591.891L421.146,591.891zM421.146,591.891',
            onclick: function()
            {
                window.open("citationtree.php?paperid="+"<?php echo($Paper_id);?>"+"&number="+"<?php echo($
                    Bas_cita_num);?>");
            },
        }
    }
},
```

The realization of the Big citation tree page is not the same as the small citation tree or reference tree. Put the mouse on the tree ,the title and the publish year will be shown too. We have to emphasize that the tree is large so the visitors might have to wait for a minute before the tree can be seen.

6 About Conference Page

The conference page presents the information and basic condition of one specific conference.
Click any name of conference in other pages to see its ocnference page.
I will introduce this page as separated parts.

6.1 Title

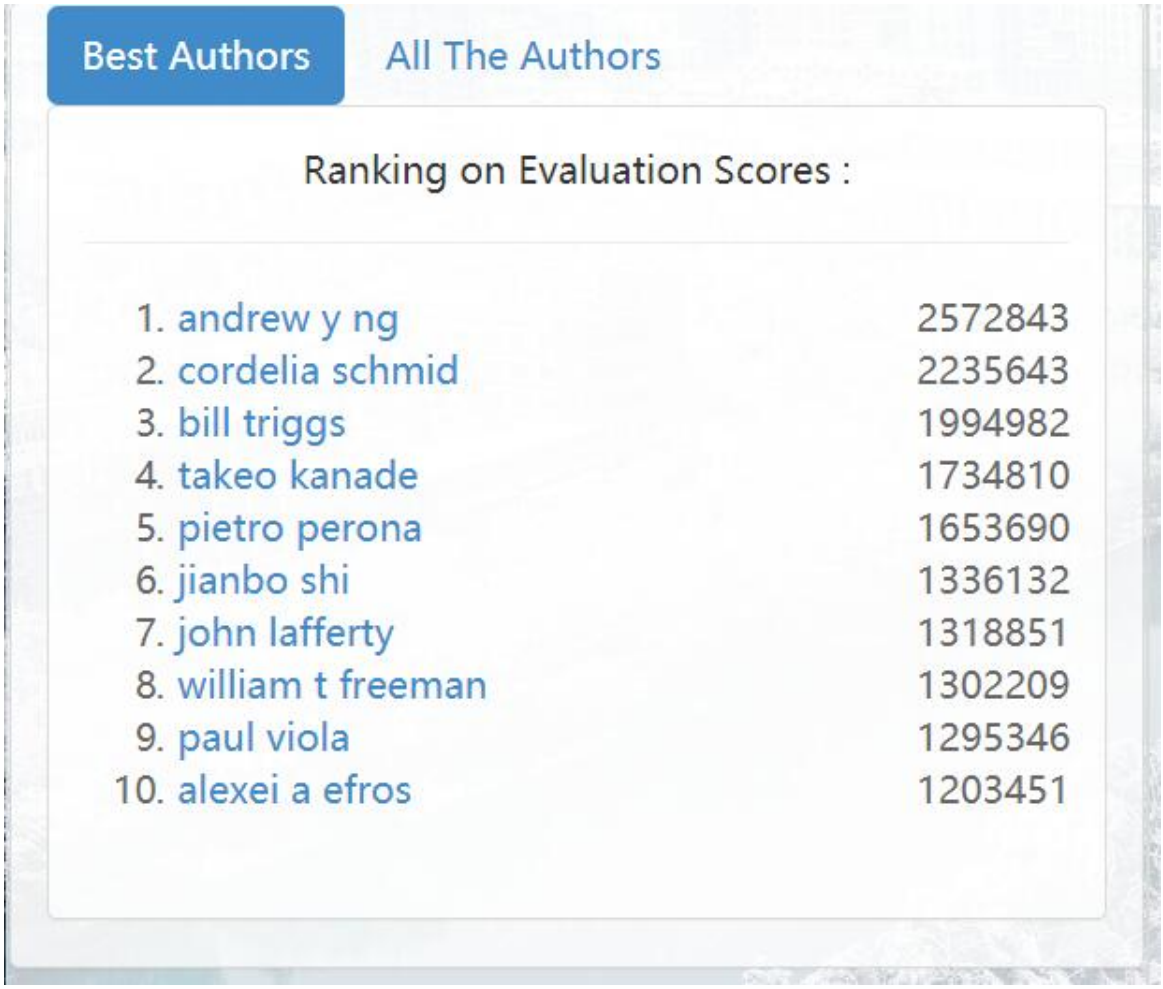
The Confernce Name

Here I use abbreviation for the conference name so that when focusing on it we can see its full name.

6.2 List of Authors

In this part I list the authors in two ways, and it can be switched by the buttons above:

- Here I rank the ten best authors according to the scores assigned by my teammate.



- Here I list all the authors by the number of papers.

- The source code:

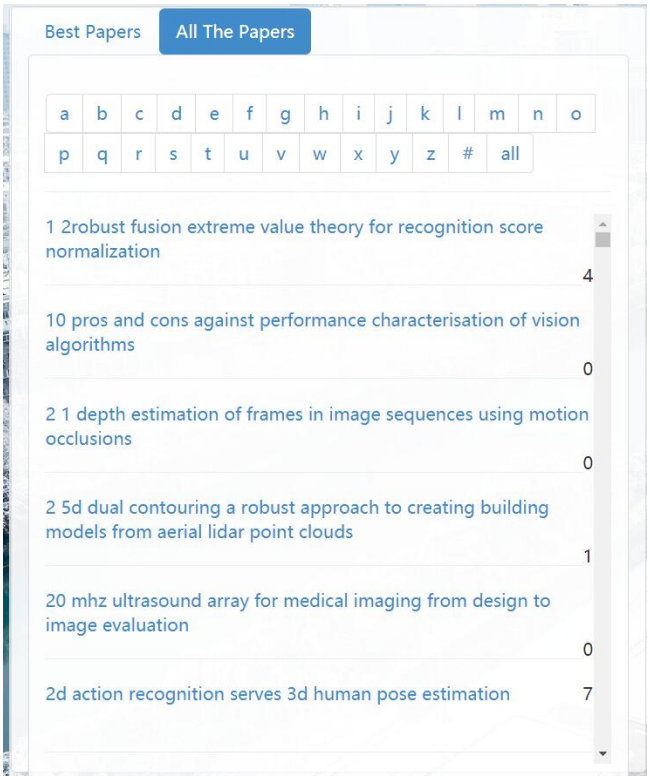
```
<div class="panel panel-default">
    <div class="panel-body" style='height:346px;'>
        <div id="tab-nav1">
            Ranking on Evaluation Scores :<hr>
        </div>
        <div id="tab-nav2" hidden='hidden'>
            Order by Article Counts :<hr>
        </div>
        <div>
        <?php
        echo "<ol class='ol' id='lists' style='height:250px;'>";
        if($result2[0][1]){
            foreach($result2 as $paper)
            {
                echo "<li><a target='_blank' href='$filelink?value=$conferencename $paper[0]'>$paper[0]</a><scan class='pull-right'>$paper[1]</scan></li>";
            }
        }else{
        echo "<li class='ln-no-match listNavHide'>No matching result...</li>";
        }
        echo "</ol>";
        echo "<div hidden='hidden' id='lists2' data-spy='scroll' data-target='#navbar-example' data-offset='0' style='height:250px;overflow:auto; position: relative;'>";
        foreach($authornames as $author=>$papers)
        {
            echo "<a target='_blank' href='$filelink?value=$conferencename $paper[0]' class='pull-left'>$author</a><scan class='pull-right'>$papers</scan></br>";
        }
        echo "</div>";
        ?>
        </div>
    </div>
</div>
```

When collecting the infomation, I use a lot MySQL inquiring statements. And they are used in most of other parts. So I show them **together** here:

```php
$conferenceid=$_GET["conferenceid"];
$link = mysqli_connect('db.lifanz.cn:3306','ee101_user','ee1012019','ee101');
if (!$link) {die("连接错误: " . mysqli_connect_error()); }
else {
$result = mysqli_query($link, "SELECT ConferenceName FROM `conferences` where ConferenceID='$conferenceid'");
$conferencename = mysqli_fetch_array($result)["ConferenceName"];
if ($conferencename){
echo "<h1>The Conference :  ";
echo "<abbr title=\"$abbrs[$conferencename]\">$conferencename</abbr></h1>";
$result = mysqli_query($link, "SELECT G.AuthorName FROM `authors` G
    inner join (
        SELECT E.AuthorID FROM `paper_author_affiliation` E inner join (
            SELECT PaperID FROM `papers` WHERE ConferenceID='$conferenceid'
        ) F on E.PaperID=F.PaperID
    ) B on G.AuthorID = B.AuthorID"
);
$result1 = mysqli_fetch_all($result);
//var_dump($result1);
$authornames = array();
foreach($result1 as $author)
{
    if (array_key_exists($author[0],$authornames))
    {
        $authornames[$author[0]]+=1;
    }
    else $authornames[$author[0]]=1;
}
arsort($authornames);
//var_dump($authornames);
$result_ = mysqli_query($link, "SELECT C.AuthorName,Score from `authors` C inner join (
        SELECT G.AuthorID,Score FROM `author_score` G
                    inner join (
                        SELECT E.AuthorID FROM `paper_author_affiliation` E inner join (
                            SELECT PaperID FROM `papers` WHERE ConferenceID='$conferenceid'
                        ) F on E.PaperID=F.PaperID
                    ) B on G.AuthorID = B.AuthorID
    ) D on C.AuthorID = D.AuthorID  group by AuthorName order by Score desc limit 0,10"
);
$result2 = mysqli_fetch_all($result_);
//var_dump($result2);
$result3 = mysqli_query($link, "SELECT Title, refcount, PaperPublishYear From (
    SELECT PaperID,Title,PaperPublishYear FROM `papers` WHERE ConferenceID='$conferenceid'
) A inner join `paper_count` B on A.PaperID=B.PaperID order by Title");
$result3 = mysqli_fetch_all($result3);
//var_dump($result3);
$titles = array();
$years = array();
foreach($result3 as $title)
{
    if (array_key_exists($title[0],$titles)) $titles[$title[0]]+=$title[1];
    else $titles[$title[0]]=$title[1];
    if (array_key_exists($title[2],$years)) $years[$title[2]]+=1;
    else $years[$title[2]]=1;
}
arsort($titles);
ksort($years);
$result4 = mysqli_query($link,"SELECT H.AuthorName,I.AuthorName as ReferenceAuthorName FROM (
    SELECT G.AuthorName,F.RefAuthorID FROM (
    SELECT E.AuthorID as RefAuthorID,C.authorID FROM (
            SELECT A.PaperID,B.AuthorID FROM (
                SELECT PaperID FROM `papers` where ConferenceID='$conferenceid'
            ) A inner join `paper_author_affiliation` B on A.PaperID=B.PaperID
        ) C inner join (
            SELECT D.PaperID,D.ReferenceID,C.AuthorID FROM (
                SELECT A.PaperID,B.AuthorID FROM (
                    SELECT PaperID FROM `papers` where ConferenceID='$conferenceid'
                ) A inner join `paper_author_affiliation` B on A.PaperID=B.PaperID
            ) C inner join `paper_reference` D on C.PaperID=D.ReferenceID
        ) E on E.PaperID=C.PaperID
    ) F inner join `authors` G on F.AuthorID=G.AuthorID
    ) H inner join `authors` I on H.RefAuthorID=I.AuthorID");
$relations = mysqli_fetch_all($result4);
```

6.3 List of Papers

This part is similar to the former part. I list the best authors according to their times of being referred to. And by clicking buttons, it can show all the authors order by the first letter.



The second one has the categorizing fucntion.
The basic idea is to assign IDs to each term corresponding to their first letter, and when categorized, it just need to hide others.
And this way is also used to the tab buttons.

```
}
//论文列表切换
function a_firsttab(){
    $("#a-lastTab").removeClass('active');
    $("#a-firstTab").addClass('active');
    $("#a-tabnav2").hide();
    $("#a-tabnav1").show();
    $("#titles2").hide();
    $("#titles1").show();
}
function a_lasttab(){
    $("#a-firstTab").removeClass('active');
    $("#a-lastTab").addClass('active');
    $("#a-tabnav1").hide();
    $("#a-tabnav2").show();
    $("#titles1").hide();
    $("#titles2").show();
}
```

```
<?php
$letters = array();
for ($t='a';$t<'z';$t++) $letters[]=$t; $letters[]='z';
foreach($letters as $x)
{
    echo "function $x()
    {
        $('.all').hide();$('.$x').show();
    }";
}
?>
function all()
{
    $('.all').show();
}
function other()
{
    $('.all').hide();
    $('.other').show();
}
```

6.4 Word Cloud for Titles

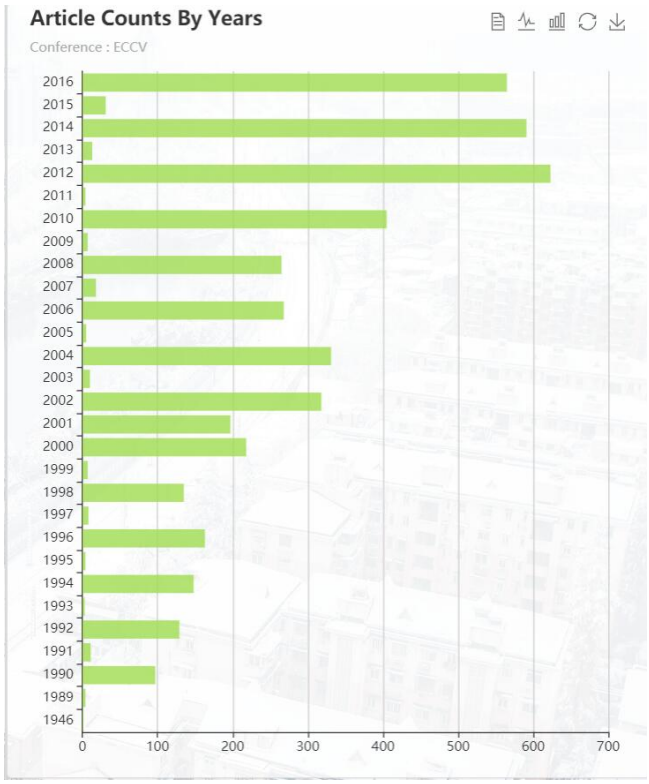
This word cloud is generated statically on the website "wordart.com".

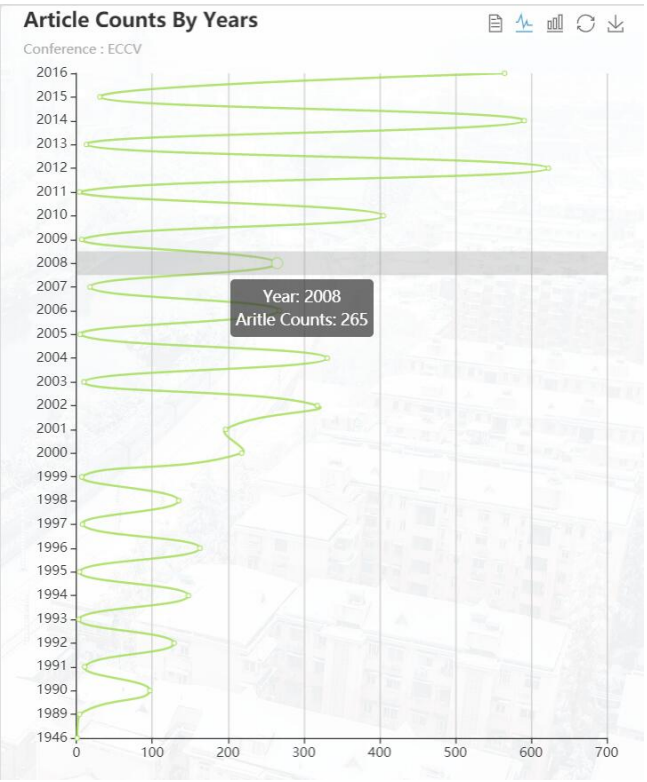
And each term in this cloud has a hyperlink to its search result page.

6.5 Chart for Article Counts

This chart is realized by the Echarts API. Actually I do not need to set anything but the data by myself.
This histogram (or line graph when switched) is to present the number of papers for each year. The tools are supported by Echarts.

The functions are as follows:





6.6 Graph for Authors

The final part which is located in the bottom of the page is about the reference realtions between the authors.

Altogether there are thousands of authors and hundreds of thousands of relationships in one conference.
However, most of the authors have only arbitrary relationships with others, which is of little value for us.

So I only take 100 authors to draw this graph. Also it's almost the maximum because of the calculation performance of the browser.

The setting for points:

```
data : [ /*{id : 0, category : 0,   name : 'aaa',   symbol : 'diamond', value : 20,nsymbolSize : 10},*/
<?php
    $author_ref = array();
    $ns=0;
    $ls=0;
    foreach($relations as $author)
    {
        if (array_key_exists($author[1],$author_ref)) $author_ref[$author[1]]+=1;
        else $author_ref[$author[1]]=1;
    }
    arsort($author_ref);
    $author_ref = array_slice($author_ref,0,100);
    foreach($author_ref as $author=>$s)
    {//$ns+=1;
        $size = intval(12+log($author_ref[$author],2));
        //if ($ns==100) break;
        echo "{
            id : '$author',
            category : 0,
            name : '$author',
            symbol : 'circle',
            value : $author_ref[$author],
            symbolSize : $size,
        },";
    }
?>
],
```

The setting for lines:

```
links : [ /*{source : 1, target : 0 },*/
<?php
    foreach($relations as $author)
    {
        if (array_key_exists($author[0],$
author_ref) && array_key_exists($author[1],$author_ref))
        {
            $ls+=1;
            echo "{
                source : '$author[0]',
                target : '$author[1]'
            },";
        }
    }
?>
]
```

The graph has two styles — circular layout or force orientation.