EE101 Final Project Report
Team 32

Litao Zhou, Shaoheng Fang, Shiwen Dong, Hongbo Yang

June 23, 2020

Contents

Preface

Who we are

We are a group of freshman students at SJTU working for the 2019 Spring EE101 course final project. We are heading for a website which allows users to search for papers and learn more about the details from our webpage and visualized charts. Our website is a local one based on a local MySQL database and a local solr search engine. The programming and coding languages we've used in this project involves Python, PHP, javascript, HTML and CSS.

What we have

Start from the index page, by typing the keywords you want to search about, you can access the search page. And then you can jump between all the information pages as you will.

Based on the type of data and the way we present them, the contents of our web fall into 5 parts, namely search (result), papers, authors, affiliations and conferences. Every section is made up of several subpages demonstrating different functions. The main functions and features of every section are listed as follows. The name behind every row represents the person who mainly takes charge, but it should be decleared that all the team members have been helping address each other's probelms all the way through the project, and must have somehow contributed to the part that he is not in charge of.

index.php

- An Integrated Search Bar *by Shaoheng Fang*
- A 3D-Graph Showing the Distribution of Years and Conferences *by Shaoheng Fang*
- The Transplantation & modification of an online Template *by Shaoheng Fang*

Figure 1: General Structure of our Web

Search Section

- Search Results Formatting *by Litao Zhou*
- Visulized Graphs *by Shaoheng Fang*
  - Publish Year
  - Conference
  - Top Authors
- Leaf Flipping in Search Results *by Shiwen Dong*

Paper Section

- Paper Information *by Litao Zhou*
- Paper Relation Chart *by Litao Zhou*
- Leaf Flipping in multiple Pages *by Shiwen Dong*
- Paper Reference/Citations/Recommendation *by Hongbo Yang*

Author Section

- Author Information *by Litao Zhou*
- Visulized Graphs *by Shaoheng Fang*
  - Publish Year (Pie Chart & Bar Chart)
  - Conference (Pie Chart & Bar Chart)
- Leaf Flipping in Author Publications *by Shiwen Dong*

Affiliation Section

- Affiliation Author List & Conference List *by Litao Zhou*
- Visulized Graphs *by Shaoheng Fang*
  - Publish Year
  - Top Author by Reference
  - Top Author by Publication
- Leaf Flipping in Author List and Conference List *by Shiwen Dong*

Conference Section

- Paper Table *by Litao Zhou*
- Paper Publish Year Graph *by Shaoheng Fang*
- Author Relation Big Graph by Gephi *by Hongbo Yang*
- Leaf Flipping in Paper Table *by Shiwen Dong*

How we work together

The making of a website requires frequent communication of source codes. In response, we use Github as a platform to exchange our codes. At the beginning of our project, we took some time to learn how Git works and created a uniform database and solr engine to make our codes work on everyone's computer. Then every member will be working and making progress on his own branch. We can use the Git system to merge our progress onto the master branch when necessary, and distribute the new version to everyone's branch. With the help of the Git system, we no longer have to bother comparing different codes written by multiple collaborators or having trouble with the inconsistency in others' codes. The network graph was derived from our Github repository homepage, where our commits and branches are displayed.

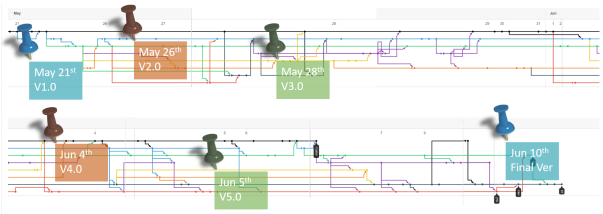


Figure 2: Our Developing Story

How to Check our Work

To see our source codes, you can visit our github repository homepage (https://github.com/ltzone/EE101Lab) and clone our repository down to your computer. You may also find the LaTeX source codes of this report in our repository. In order to make our website work on your computer, you have to follow our guide in the configguide.txt. This guide will help you initialize your MySQL database and Solr engine and write all the information into them. You may also check our presentation slides to see the detail of our developing story.

Chapter 1

Enrich the Contents

1.1 Paper & Conference Pages

These parts are required as a necessary part of final lab. We want to display useful information to users. So we added functions to search for more detailed relavant messages. According to the characteristics of each field, we used different methods to show information.

1.1.1 Description

In our final version, there are 5 concrete pages about paper information and 3 about conferences. We tried to display the information in all directions. There are blanks to show basic information, E-charts of papers, citations and references and relavant recommedations.

1.1.2 Paper: Basic Information

In this page, we show basic messages of papers which is from MySQL table 'papers' and ' paper_author_affiliation'. And we offer the function to jump to Conference, Author, Affiliation Pages and Acemap.

The codes are listed below. (Realization of Paper Charts will be introduced in chapter visualization.)

First, we have PaperID through "Get" method, then we can find result in MySQl. If result exists, we echo out all fields of paper-relavant information. (Remember that AuthorName is in the format of Arrays.)

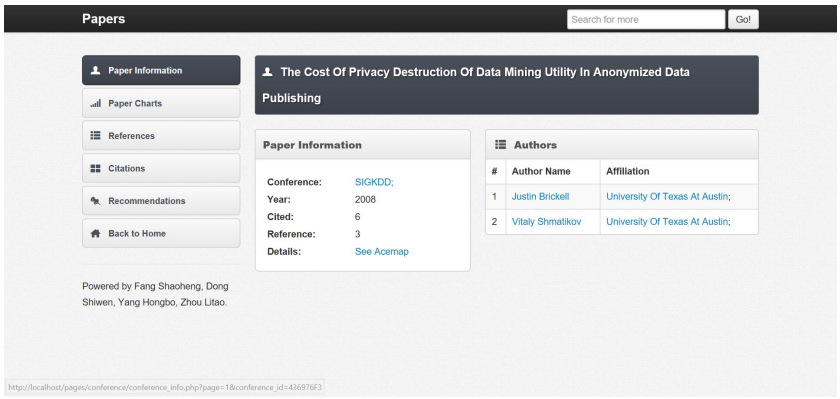


Figure 1.1: Page Example

```
#查找paper对应的Author, conference, year
$result = mysqli_query($link, "SELECT PaperID from paper_author_affiliation where PaperID='$paper_id'");
if ($result) {
    $row = mysqli_fetch_array($result);
        echo "<tr>";
        $paper_id = $row['PaperID'];
        $paper_info = mysqli_fetch_array(mysqli_query($link, "SELECT ConferenceID , PaperPublishYear from papers where PaperID='$paper_id'"));
        if ($paper_info){
            echo "<table>";
            echo "<tr><td width = '120'><b> Authors: </b></td><td>";
            $author_info = mysqli_query($link, "SELECT A.AuthorID, AuthorName FROM paper_author_affiliation A LEFT JOIN authors B ON A.AuthorID =
                while ($author_row = mysqli_fetch_array($author_info)){
                $author_name = $author_row['AuthorName'];
                $author_another_id = $author_row['AuthorID'];
                $author_name2 = ucwords($author_name);
                $author_another_id2 = ucwords($author_another_id);
                echo "<a href=\"author.php?page=1&author_id=$author_another_id2\">$author_name2; </a>";
            }
            echo "</td></tr>";
            echo "<tr><td><b> Conference: </b></td><td>";

            $conference_id = $paper_info['ConferenceID'];
            $year= $paper_info['PaperPublishYear'];

        }
```

Figure 1.2: Code Example

```
$conference_row = mysqli_fetch_array(mysqli_query($link, "SELECT ConferenceName from conferences WHERE ConferenceID = '$conference_id'"));
$conference_name = $conference_row['ConferenceName'];
$conference_name2 = ucwords($conference_name);
echo "<a href=\"conference.php?page=1&conference_id=$conference_id\">$conference_name2; </a>";
echo "</td>";
echo "</tr>";
echo "<tr><td><b> Year: </b></td><td>";
echo $year;
echo "</td></tr>";

echo "<tr><td><b> Cited: </b></td><td>";
$cited_count = mysqli_fetch_array(mysqli_query($link,"SELECT count(*) FROM paper_reference2 WHERE ReferenceID = '$paper_id';"));
echo ($cited_count[0]);
echo "</td></tr>";


echo "<tr><td><b> Reference: </b></td><td>";
$refer_count = mysqli_fetch_array(mysqli_query($link,"SELECT count(*) FROM paper_reference2 WHERE PaperID = '$paper_id';"));
echo ($refer_count[0]);
echo "</td></tr>";
```

Figure 1.3: Code Example

1.1.3 Paper: Citation and Reference

This part is quite similar to paper information part(just add a while loop). Before doing this, we should know what is thrown into MySQL. In the fuction

above, that is PaperID, and in Reference part, it's ReferenceID while in Citation part, we swap paper and reference.

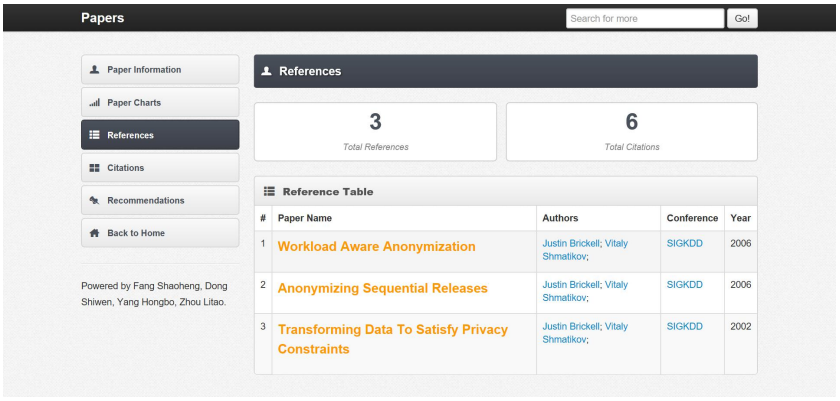


Figure 1.4: Page Example

Here we give code: (just the first part, the followed part has been already introduced.)

```
$result = mysqli_query($link, "SELECT PaperID from paper_reference2 where ReferenceID='$paper_id'");
if ($result->num_rows) {
    echo "<div class='paperlis'>";
    echo "<h1 style=\"font-family:Arial Black\">被引用文章</h1>";
    while ($row = mysqli_fetch_array($result)) {
        $paper_id_ref = $row['PaperID'];
        $paper_info = mysqli_fetch_array(mysqli_query($link, "SELECT Title, ConferenceID from papers where PaperID='$paper_id_ref'"));
```

Figure 1.5: Code Example

This is Citation Page example:

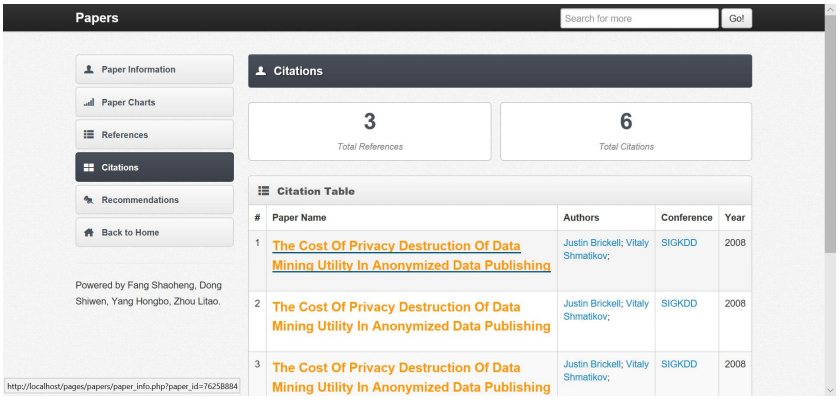


Figure 1.6: Page Example

Here we give code:

```php
#reference查找
#增加查询reference结果为空的条件判断
$result = mysqli_query($link, "SELECT ReferenceID from paper_reference2 where PaperID='$paper_id'");
if ($result->num_rows) {
    echo "<div class='paperlis'>";

    echo "<h1 style=\"font-family:Arial Black\">引用文章</h1>";
    while ($row = mysqli_fetch_array($result)) {
        $paper_id_ref = $row['ReferenceID'];
        $paper_info = mysqli_fetch_array(mysqli_query($link, "SELECT Title, ConferenceID from papers where PaperID='$paper_id_ref'"));
```

Figure 1.7: Code Example

1.1.4 Paper: Recommendation

We came up with 2 method to recommend papers:

1) Recommend by relavant title

This method relies on solr. In solr, field Title is stored as 'text-en' format. So we can do vigrous search to match related papers First, we should search in MySQL to find Title (at the beginning, we have paper_id). As the search result is in the format of (" xxxxx "), we should process it to xxxxx to transmit to solr-url. Specificly, we use (var:)paper_title4=substr((var:)paper_title3,3,-2);

```php
$paper_id = $_GET["paper_id"];
$paper_ti=mysqli_fetch_array(mysqli_query($link, "SELECT Title from papers where PaperID='$paper_id'"));
$paper_title3=$paper_ti['Title'];
    $ch = curl_init();
$timeout = 5;
//echo $paper_title3;
$paper_title4=substr($paper_title3,3,-2);
$query = urlencode(str_replace(' ', '+', $paper_title4));
$url = "http://localhost:8983/solr/FINAL/select?q=PaperName%3A".$query."&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);
```

Figure 1.8: Code Example

Then similarly, we can echo result out. (See Figure 1.1.4 on the next Page)

2) Recommend by relavant Author

This method relies on MySQL. The central sentence of this part is the MySQL join-query sentence below:

```
1   SELECT d.PaperID, Title from
2   (SELECT PaperID From (SELECT AuthorID FROM paper_author_affiliation a where a.PaperID = '0000E395') b
3   inner join paper_author_affiliation c on b.AuthorID = c.AuthorID group by PaperID) d
4   inner join papers on d.PaperID = papers.PaperID
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PaperID	Title
0000E395	a general semantic analyser for data base access
094853E6	the cambridge university multimedia document r…
5B0003E0	effects of out of vocabulary words in spoken do…
7FF5CF5A	retrieving spoken documents by combining multi…

Figure 1.10: Code Example

Here we give complete codes:

```php
// 显示搜索结果的分区
    if ($result['response']['numFound']>0){
        echo "<div class='paperlis'>";
        echo "<h1 style=\"font-family:Arial Black\">相关文章内容</h1>";
        foreach ($result['response']['docs'] as $paper) {

            $paper_id = $paper['PaperID'];
            $papername2 = ucwords($paper['PaperName']);
            echo "<a href=\"paper.php?paper_id=$paper_id\"><h3>$papername2</h3></a>";
            echo "<table>";
            echo "<tr><td width = '120'><b> Authors: </b></td><td>";

            foreach ($paper['AuthorName'] as $idx => $author) {
                $author_id = substr($paper['AuthorID'][$idx],2,-3);
                $author2 = ucwords($author);
                echo "<a href=\"author.php?page=1&author_id=$author_id\">$author2</a>";
                echo "; ";
            }
            echo "</td></tr>";
            echo "<tr><td><b> Conference: </b></td><td>";
            $conference_id =$paper['ConferenceID'];
            $conference = $paper['ConferenceName'];
            echo "<a href=\"conference.php?page=1&conference_id=$conference_id\">$conference</a>";
            echo "; ";
            echo "</td></tr>";
            echo "</table>";
            echo "<hr>";
```

Figure 1.9: Code Example

```php
//通过作者推荐文章
//
//
//

$paper_id=$_GET["paper_id"];
$result = mysqli_query($link, "SELECT d.PaperID, Title from (SELECT PaperID From (SELECT AuthorID FROM paper_author_affiliation a where a.
var_dump($result);
if ($result) {

echo "<div class='paperlis'>";
echo "<h1 style=\"font-family:Arial Black\">相关作者的文章: </h1>";echo "<hr>";

    echo "<div class='paperlis'>";
    while ($row = mysqli_fetch_array($result)){
        $paperkkk=$relatepaper['PaperID'];
        $flag=($paperkkk!=$paper_id);
        if ($flag){
            echo "<table>";
            $papernamekkkrow=mysqli_fetch_array(mysqli_query($link, "SELECT Title From papers where PaperID='$paperkkk'"));
            $papernamekkk=$papernamekkkrow['Title'];
            echo "<a href=\"paper.php?paper_id=$paperkkk\">$papernamekkk; </a>";
```

Figure 1.11: Code Example

Here we give exammple page:

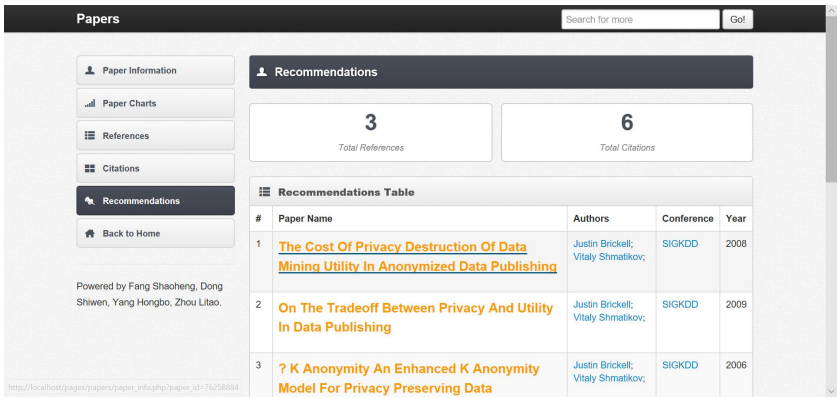


Figure 1.12: Page Example

1.1.5 Conference Information

In conference page, we showed basic information of conferences nameiy the number of authors, papers and references. Then we listed the papers that are included in the distinct conference and gave hyperlinks to other pages.

(The impletment of count function is shown in the next subsection.)

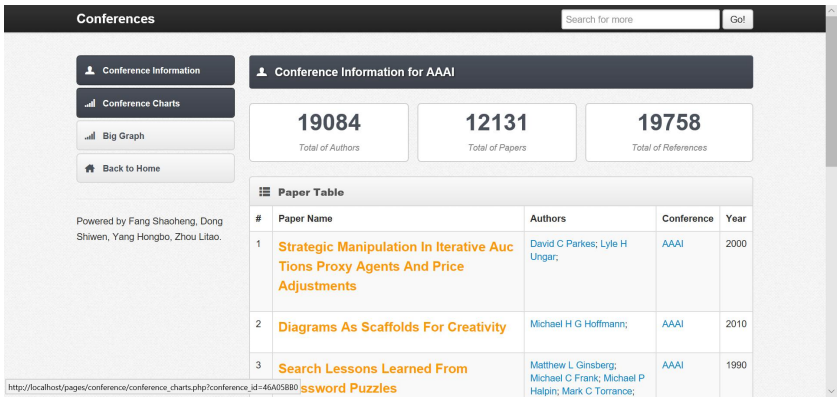


Figure 1.13: Page Example

There are 3 blanks on the right side, which show the total number of authors, papers and references. We call this “count function”. We wrote 2 versions. The first version use counting varaible in loops to count number, which is not so efficient. So we turned to another method, which will be introduced in detail in “affiliation page”.

Below these is a paper list. Here is the code to search for papers that are included in the conference, the process to achieve it is similar to that of

"author.php" in last homework. We used MySQL to find the information we want.

First, we have ConferenceID through 'Get', then we can use code:

```
$page_num=$_GET['page'];
$page_total=(integer)(($num_results+9)/10);
if($page_num>$page_total)$page_num=$page_total;

$result = mysqli_query($link, "SELECT PaperID from papers where ConferenceID='$conference_id'limit ".(($page_num-1)*10).",10 ");
// 显示搜索结果的分区
if ($result) {
    echo "<hr>";
    echo "<div class='paperlis'>";
    while ($row = mysqli_fetch_array($result)) {
        echo "<tr>";
        $paper_id = $row['PaperID'];
        $paper_info = mysqli_fetch_array(mysqli_query($link, "SELECT Title, ConferenceID, PaperPublishYear from papers where PaperID='$paper_id'"));
```

Figure 1.14: Code Example

to find PaperID. And problem comes to using PaperID to find paper information. We just follow the approach that was used in last homework, and echo the needed messages out. There is a point that should be mentioned. AuthorID field has multiple value, so we have to process like an array.

```
echo "<div class='paperlis'>";
while ($row = mysqli_fetch_array($result)) {
    echo "<tr>";
    $paper_id = $row['PaperID'];
    $paper_info = mysqli_fetch_array(mysqli_query($link, "SELECT Title, ConferenceID, PaperPublishYear from papers where PaperID='$paper_id'"));
    if ($paper_info){
        $paper_title = $paper_info['Title'];
        $conf_id = $paper_info['ConferenceID'];
        $paper_title2 = ucwords($paper_title);
        echo "<a href=\"paper.php?paper_id=$paper_id\"><h3>$paper_title2</h3></a>";
        echo "<table>";
        echo "<tr><td width = '120'><b> Authors: </b></td><td>";
        $author_info = mysqli_query($link, "SELECT A.AuthorID, AuthorName FROM paper_author_affiliation A LEFT JOIN authors B ON A.AuthorID = B.Aut

        while ($author_row = mysqli_fetch_array($author_info)){
            $author_name = $author_row['AuthorName'];
            $author_another_id = $author_row['AuthorID'];
            $author_name2 = ucwords($author_name);
            $author_another_id2 = ucwords($author_another_id);
            echo "<a href=\"author.php?page=1&author_id=$author_another_id2\">$author_name2</a>";
            echo "; ";
        }
        echo "</td></tr>";
        echo "<tr><td><b> Conference: </b></td><td>";


        $conference_row = mysqli_fetch_array(mysqli_query($link, "SELECT ConferenceName from conferences WHERE ConferenceID = '$conf_id'"));
        $conference_name = $conference_row['ConferenceName'];
        $conference_name2 = ucwords($conference_name);
        echo "<a href=\"conference.php?page=1&conference_id=$conf_id\">$conference_name2</a>";
        echo "</td></tr>";
```

Figure 1.15: Code Example

We also added E-charts and Gephi graphs in related pages to show the information of conference more clearly, which method will be discribed in detail in following chapters.

1.2 Affiliation Pages

We'd like to add a new series of pages to show affiliation information to users. Since we didn't do any previous work about affiliation information in the previous labs except that the affiliation table was input into the database, we have to write new SQL commands in search of affiliation information, and echo them out on the pages. The paper table and charts on other pages can be of help in showing the affiliation related information. Also, we have already got

the author table written in the paper information page. Generally speaking, the work here is to collect the affiliation data and arrange them in order on the pages.

1.2.1 General Description

The final version of the affiliation section include 3 concrete pages. On the affiliation_info.php page, we will give three numbers on top of the page, counting all the authors, papers, and references in the affiliation. Then the authors in the affiliation will be listed below, together with their own affiliation information (hyper-links included) and number of publications. Affiliation_paper.php shows all the papers related to the affiliation. The affiliation_charts.php, where three charts are displayed, will be reported in detail in the Statistical Graph Section. (See Figure 1.16 and Figure 1.17 on the next Page)

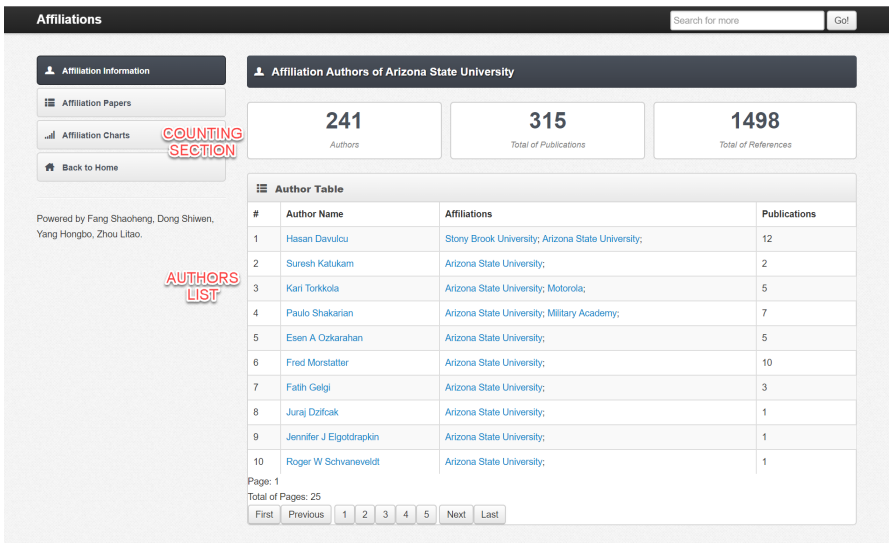


Figure 1.16: affiliation_info.php page

In fact, for the same function to be displayed on the page, we did two versions of code to implement them. The first one was very basic, just like those in the paper/author/search pages. However, since all the affiliation data are selected from the big paper_author_affiliation table, the first version didn't perform well in loading speed. Our optimization will be introduced in the Optimization Chapter.

1.2.2 Total Counts

On top of every affiliation page, we've counted all the authors, papers and references concerned. For authors and papers, we can directly count them in

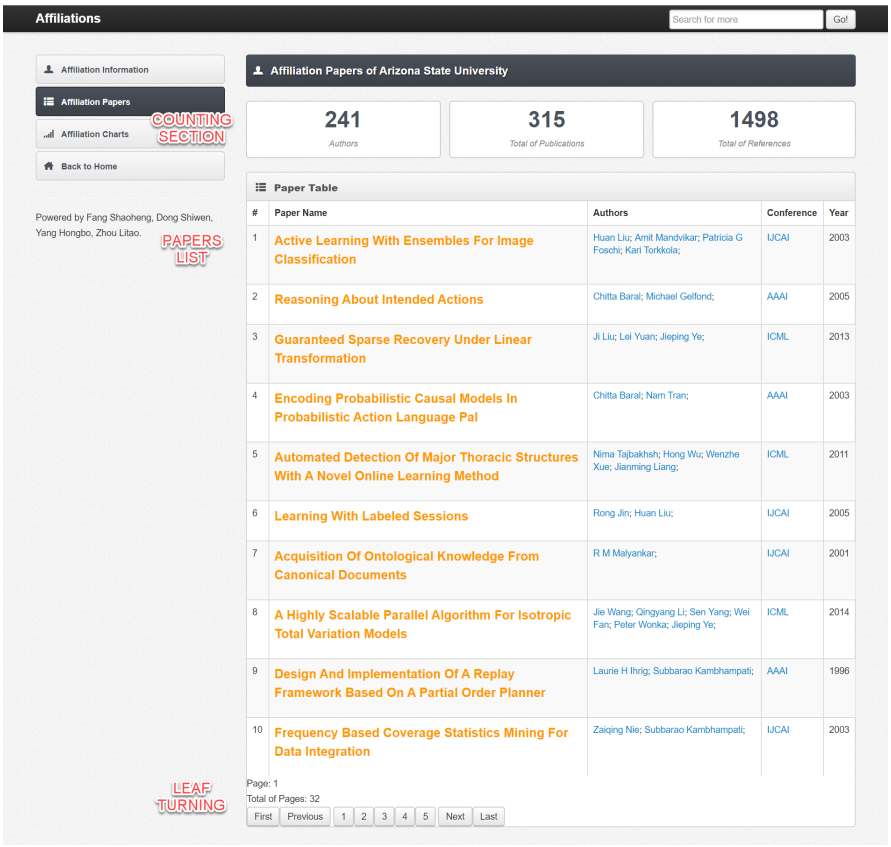


Figure 1.17: affiliation_paper.php page

the paper_author_affiliation table. For references, we first select the papers and then join the selected results to the paper_reference table in order to count the results. The MySQL commands are listed below.

```
# 查询本机构所有作者（不重复）
$result = mysqli_query($link, "SELECT AuthorID, count(distinct AuthorID) from paper_author_affiliation where AffiliationID='$
    affiliation_id' group by AuthorID");
$author_list = mysqli_fetch_all($result);
$author_count= count($author_list );
```

Figure 1.18: Count Author Commands

```
$pubresult = mysqli_query($link, "SELECT PaperID, count(distinct PaperID) from paper_author_affiliation where
    AffiliationID='$affiliation_id' group by PaperID");
echo $pubresult->num_rows;?>
```

Figure 1.19: Count Paper Commands

```php
$result =mysqli_query($link,"SELECT count(*) from (SELECT PaperID, count(distinct PaperID) from
    paper_author_affiliation where AffiliationID='$affiliation_id' group by PaperID) A INNER JOIN paper_reference2 B on
     A.PaperID = B.PaperID");
$result = mysqli_fetch_all($result)[0][0];
echo $result;
```

Figure 1.20: Count Reference Commands

Note that we've use some MySQL techniques such as DISTINCT (eliminate overlapping results) and GROUP BY (perform data counting job) in order to implement our function.

For the data display in our page, our template has already provided a well designed data container in CSS, which can list different numbers in a row, with even width. We can simply apply this pre-defined class in a way demonstrated below.

```php
<div class="stat-container">

    <div class="stat-holder">
        <div class="stat">
            <span><?php echo $author_count;?></span>
            Authors
        </div> <!-- /stat -->
    </div> <!-- /stat-holder -->
```

Figure 1.21: Use Pre-defined Class to Display Numbers

1.2.3 Authors List

The author we've found based on the given affiliaiton may have more than one affiliation where he publishes his paper. So one simple search work is not enough. Luckily, all of the data related to this problem can be selected from the paper_author_affiliation table. As a result, our first design is to first sort out all the authors where the affiliation column fits, then we search the table again for affiliation information based on the author's id, which can be realized by looping through the author array in PHP.

In fact, the first step has been done when we count the authors (See Figure 1.18), so we simply skip the first step, call the author selecting result in the counting section, and make further searching based on the previous result. During the second step, we've also used DISTINCT & GROUP BY techniques in order to get the author's affiliation data right and unrepeated.

```php
for ($i=0;$i<$author_count;$i+=1){
    $author_id = $author_list[$i][0];
    $result = mysqli_query($link, "SELECT AuthorName from authors where AuthorID='$author_id'");
    $author_name = mysqli_fetch_array($result)['AuthorName'];
    echo "<tr>";
    echo "<td>";
    echo $i+1;
    echo "</td>";
    echo "<td>"."<a href=\"../authors/author_info.php?author_id=$author_id\">".ucwords($author_name)."</a></td>";
    $Affresult = mysqli_query($link, "SELECT Affiliations.AffiliationID, Affiliations.AffiliationName from (select
        AffiliationID, count(*) as cnt from paper_author_affiliation where AuthorID='$author_id' and AffiliationID is not
        null group by AffiliationID order by cnt desc) as tmp inner join Affiliations on tmp.AffiliationID = Affiliations.
        AffiliationID");
    echo "<td>";
    if ($Affresult->num_rows!=0){
        foreach ($Affresult as $affline){
            $Affi_id = $affline['AffiliationID'];
            $Affi_name = ucwords($affline['AffiliationName']);
            echo "<a href=\"../affiliations/affiliation_info.php?affiliation_id=$Affi_id\">$Affi_name</a>;\n";}
        echo "</td>";
        }
    echo "<td>";
    $result = mysqli_query($link, "SELECT count(PaperID) from paper_author_affiliation where AuthorID='$author_id'");
    $pub_count =  mysqli_fetch_array($result)[0];
    echo $pub_count;echo "</td>";
```

Figure 1.22: Use Loop to List Author_Affiliation Information

The problem with this solution is that the searching work is too much. Imagine there are 100 authors related to one affiliation, we have to go through the big paper_author_affiliation table 100 times in order to get all the data. It turned out that it would take the webpage about half a minute to get completed loaded. The optimization work will be introduced in the Optimization Chapter.

1.2.4 Papers List

The general structure of this list is simlilar to those in citation/ reference/ author's paper list. We may just use MySQL to select all the paper's id related to the affiliation, keep this array storing the paperid we want to display, and the leave the rest of the work to the codes we've already written in the previous work. The selecting MySQL commands are listed below. Actually, just like the case in the author list section, the first step has also been completed in the counting section. (See Figure 1.19) So there are no codes left for this section to explain.

Chapter 2

Leaf Turning

Page turning is an indispensable part of web design. When there is more information to be searched and displayed, the page turning function can reduce the number of information displayed on each page, which makes the whole page look more concise and beautiful, and also optimizes the user's experience.

In this experiment, we have designed two versions of page turning function. And the detailed information of them are as follows.

2.1 Version I: by PHP parameters

In this version,we set the parameter 'page' for each page, and the show scope is determined according to the'page' parameter and each time we don't show all data. Actually, because of specific value of page, we only show 10 pieces of data.

2.1.1 Description

Now let's show the first version's page turning . We take the search results on the'search'page as an example, and the effect is shown in the following figure. (See Figure 2.1.1)

Ten pieces of information are displayed on each page. At the bottom of each page, the number of the current page, total data and total pages can be displayed. By clicking the'previous page's and 'next page's hyperlinks, the page can be turned up and down. By clicking the'first page's and 'last page's hyperlinks, the page can be turned to the first and last pages. In addition, we can also input the page number of the page we want to see in the blank and click the button 'jump to the page' to achieve the page jumping.

In this way, we can search the data spending little time . But the disadvantage of this method is that every time we turn the page, we indeed open a new page, and everying needs to be queried and shown again, even though which maybe needn't the page turning function.

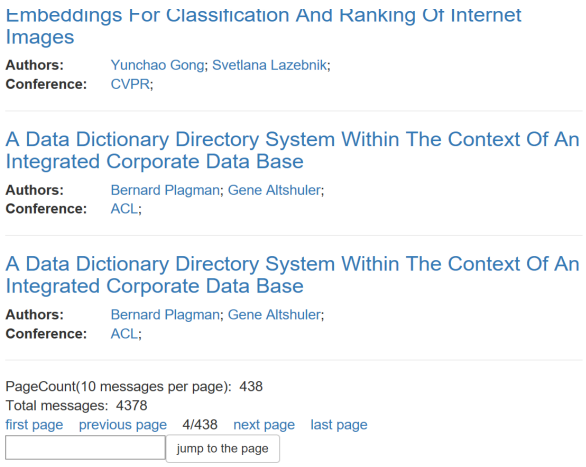


Figure 2.1: Base Version Demonstration

2.1.2 Solution

We set the parameter 'page' for each page, and the show scope is determined according to the'page' parameter to make sure that we get specific 10 pieces of data in order to achieve the page turning. Every page's query statments are different because they contain the 'page' paremeter. And the default value of the 'page' is 1.

The page turning is achieved by hyperlinks, and each time we click the specific word or input the page jumping number, we will turn to a new page with different papremeter 'page'.

2.1.3 Source Codes

Now we still take the search page as an example to present the page turning. The codes are as follows.

```
$page_num=$_GET['page'];
if(!$page_num)$page_num=1;
if($page_num<0)$page_num=1;
$num_results =$result['response']['numFound'];
$page_total=(integer)(($num_results+9)/10);
if($page_num>$page_total)$page_num=$page_total;
```

Figure 2.2: Page Parameter

At first, we get the paremeter 'page' sent from other pages. Before we search information , we should know the number of total information we get and figure out the number of total pages which depends on how many pieces of data we

```php
echo "<div class='paperlis'>";

$num_results =$result['response']['numFound'];
$page_total=(integer)(($num_results+9)/10);
if($page_num>$page_total)$page_num=$page_total;
if($page_num==$page_total)$l=$num_results;
else $l=($page_num-1)*10+10;



for ($i=($page_num-1)*10;$i<$l;$i++) {
    $paper=$result['response']['docs'][$i] ;
    $paper_id = $paper['PaperID'];
    $papername2 = ucwords($paper['PaperName']);
    echo "<a href=\"paper.php?paper_id=$paper_id\"><h3>$papername2</h3></a>";
    echo "<table>";
    echo "<tr><td width = '120'><b> Authors: </b></td><td>";
    foreach ($paper['AuthorName'] as $idx => $author) {
        $author_id = substr($paper['AuthorID'][$idx],2,-3);
        $author2 = ucwords($author);
        echo "<a href=\"author.php?page=1&author_id=$author_id\">$author2</a>";
        echo "; ";
    }
    echo "</td></tr>";
    echo "<tr><td><b> Conference: </b></td><td>";
    $conference_id =$paper['ConferenceID'];
    $conference = $paper['ConferenceName'];
    echo "<a href=\"conference.php?page=1&conference_id=$conference_id\">$conference</a>";
    echo "; ";
    echo "</td></tr>";
    echo "</table>";
    echo "<hr>";

}
```

Figure 2.3: Show

want to show in a page. Then we use 'for' circle statement to show the data . The codes are shown in the figure'page parameter'and the figure 'show'.

When setting the hyperlinks, what matters is that we should send the parameter 'keyword' besides the parameter 'page'. Another important thing is that sometimes some hyperlinks needn't be use. For example, if we are at the first page, the hyperlinks of the statement 'first page' is unnecessary.So we should write some specific 'if' statement to judge if we should use the hyperlinks. (See 2.4 on the next Page)

```php
echo "<form  action=\"search.php\"><div style=\"text-align:left;\">";
echo "<input type=\"integer\"  id=\"page\" name=\"page\">";
echo "<input name='keyword' type='hidden' id='keyword' value=$keyword>";
echo "<button type=\"submit\" class=\"btn btn-default\">jump to the page</button></div></form>";
echo "</div>";
```

Figure 2.5: Paper Jumping

When it comes to the page jumping, what should be mentioned is that to reduce the error such as the number user entered is out of the limitation of the actual pages, we let the actual page you jump to is always the last page if you input a number that is greater than the number of total pages.

```php
// 翻页模块
echo '<p>PageCount(10 messages per page):  '.$page_total.'    </p>';
echo '<p>Total messages:  '.$num_results.'</p>';
if($page_total>$page_num )
{
    if($page_num>1)
    {
        echo '<a href="./search.php?page=1&keyword='.($keyword).'">first page   </a>    ';
        echo '<a href="./search.php?page='.($page_num-1).'&keyword='.($keyword).'"> previous page   </a>';
        echo " ".$page_num."/".$page_total."    ";
        echo '<a href="./search.php?page='.($page_num+1).'&keyword='.($keyword).'">next page   </a>';
        echo '<a href="./search.php?page='.($page_total).'&keyword='.($keyword).'">    last page</a>';
    }
    else
    {
        echo 'first page     ';
        echo ' previous page   ';
        echo " ".$page_num."/".$page_total."    ";
        echo '<a href="./search.php?page='.($page_num+1).'&keyword='.($keyword).'">next page   </a>';
        echo '<a href="./search.php?page='.($page_total).'&keyword='.($keyword).'">    last page   </a>';
    }
}
else
{
    if($page_total==1)
    {
    echo 'first page       ';
    echo ' previous page   ';
    echo " ".$page_num."/".$page_total."    ";
    echo 'next page   ';
    echo 'last page   ';
    }
    else
    {
        echo '<a href="./search.php?page=1&keyword='.($keyword).'">first page   </a>    ';
        echo '<a href="./search.php?page='.($page_num-1).'&keyword='.($keyword).'"> previous page   </a>';
    echo " ".$page_num."/".$page_total."    ";
    echo 'next page   ';
    echo 'last page';
    }
}
```

Figure 2.4: Hyperlinks

2.1.4 Demonstration

An example is demonstrated as follows.

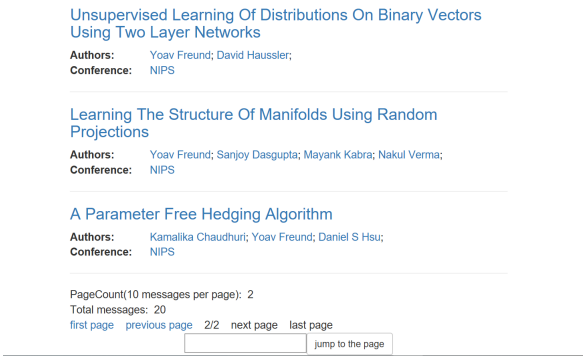


Figure 2.6: AUTHOR EXAMPLE

This is the last page of the author page.

2.2 Version II: by jQuery

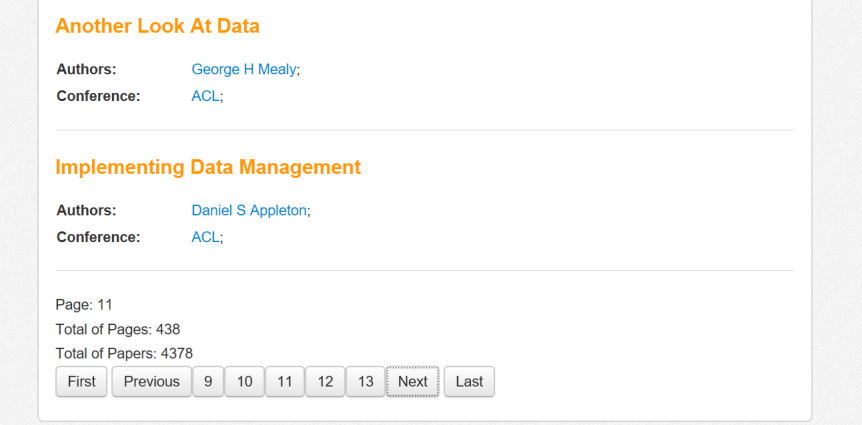
In the first version of paging design, only PHP code is used, and the use of the 'Page' parameter means that the page would be reloaded every time the page was turned. Obviously, all that needs to be refreshed is the table that shows the data, and the rest of the page will remain unchanged. If we can only refresh the data table every time you turn the page, and keep the other parts unchanged, then we can further improve the efficiency of the web page.

With this idea in mind, we started the second version of page turning design.

2.2.1 Description

First, let's show the second version's page turning . We take the search results on the'search'page as an example, and the effect is shown in the following figure.

Ten pieces of information are displayed on each page. At the bottom of each page, the number of the current page and total pages can be displayed. By clicking the'previous'and'next' buttons, the page can be turned up and down. By clicking the'first'and'next' buttons, the page can be turned to the first and last pages. In addition, we can also click the button corresponding to the page numbers before or after the current page with a total of five pages.



Because of the use of'jQuery'to achieve page turning , after a load, we can quickly change the page without refreshing, and after the page is turned, we still stay in the previous position, instead of moving back to the top of the page, which makes the page turning function more convenient and fast.

2.2.2 Solution

Unlike the first version where the search scope is determined according to the'page' parameter and each time we don't search all data, in this version, we can search all the results at one time and divide them into many parts. Each

page turn shows the information of one part and hides the information of the rest. That is to say, we can divide the searched information into different'divs', add page turning function on the button, and change the display style of different divs when clicking.

Similarly, we can divide the buttons corresponding to different page numbers into many divs, hide part and display part when clicking, and then we can achieve the page turning without refreshing.

2.2.3 Source Codes

Now we still take the search page as an example to present the page turning.

```
<body>
<script TYPE="TEXT/JAVASCRIPT" src="http://localhost/js/jquery-1.7.2.min.js"></script>
```

Figure 2.7: Premise

At the beginning, we should import the 'jquery'. The codes are shown in the figure 'premise'.

```
$num_results =$result['response']['numFound'];
$totalpage=(integer)(($num_results+9)/10);
$page=1;

//分割div

for ($i=1;$i<=$num_results;$i++) {
    if($i%10==1){echo"<div id=\"$page\"style=\"display:none\">";}
    $paper=$result['response']['docs'][$i-1] ;
    $paper_id = $paper['PaperID'];
    $papername2 = ucwords($paper['PaperName']);
    echo "<a href=\"../papers/paper_info.php?paper_id=$paper_id\"><h3>$papername2</h3></a>";
    echo "<table>";
    echo "<tr><td width = '120'><b> Authors: </b></td><td>";
    foreach ($paper['AuthorName'] as $idx => $author) {
        $author_id = substr($paper['AuthorID'][$idx],2, -3);
        $author2 = ucwords($author);
        echo "<a href=\".\..\authors\author_info.php?page=1&author_id=$author_id\">$author2</a>";
        echo ", ";
    }
    echo "</td></tr>";
    echo "<tr><td><b> Conference: </b></td><td>";
    $conference_id =$paper['ConferenceID'];
    $conference = $paper['ConferenceName'];
    echo "<a href=\".\..\conference\conference_info.php?page=1&conference_id=$conference_id\">$conference</a>";
    echo ", ";
    echo "</td></tr>";
    echo "</table>";
    echo "<hr>";

    if($i%10==0&&$i>1){echo"Page: ";echo$page;echo"<br>";echo"  Total of Pages: ";echo$totalpage;echo"<br>";echo"Total of Papers: ";
        echo$num_results;$page+=1;echo"</div>";}
    else if ($i==$num_results){echo"Page: ";echo$page;echo"<br>";echo"  Total of Pages: ";echo$totalpage;echo"<br>";echo"Total of
        Papers: ";echo$num_results;echo"</div>";}

}
```

Figure 2.8: Dividing divs

Before we divide all information into different divs, we should know the number of total information we get and figure out the number of total pages which depends on how many pieces of data we want to show in a page. Then we use 'for' circle statement to divide the data into many divs. Every div's style is set as 'hidden' at first by using the statement 'display:none'. And at the end of each div, we show the number of total pages and the current page.The codes are shown in the figure'Dividing divs'

```
            echo'
<script type="text/javascript">
var now=1;
var totalpage ='; echo $totalpage ;echo';
 document.getElementById("1").style.display="";
</script>';
```

Figure 2.9: Start

The codes in the figure 'Start' are to send the variable 'page' to the javascript and change the first page's display style from hidden into shown.

```
echo"<div>";
echo"<style type=\"text/css\">

.button {

    display: inline-block;
}</style>";
echo"<div class=\"btn-group\">
<div class=\"button\"><button type=\"button\" class=\"btn btn-default \"id=\"but1\">First</button></div>

<div class=\"button\"><button type=\"button\" class=\"btn btn-default \"id=\"but3\">Previous</button></div>";

for($j=1;$j<=$totalpage;++$j){
    $jj=(string)$j;
    echo"<div class =\"button\"><button type=\"button\" class=\"btn btn-default \"id=\"b$jj\">$jj</button></div>";}

echo"<div class=\"button\"><button type=\"button\" class=\"btn btn-default \"id=\"but2\">Next</button></div>
<div class=\"button\"><button type=\"button\" class=\"btn btn-default \"id=\"but4\">Last</button></div></div>";
```

Figure 2.10: Buttons

Then as the figure 'Buttons' shows, we create four buttons 'first','previous','next' and 'last' which are used to realize the primary function of page turning. Then we create a corresponding button for each div, all of which are in the different divs, whose display style can also be changed by clicking. To make them look neater, we set their display style as 'inline block', which make sure that all buttons are shown in one line.

```
            echo'
<script type="text/javascript">
 $(document).ready(function(){
$("#but1").click(function(){
    document.getElementById(now.toString()).style.display="none";
    document.getElementById("1").style.display="";
    now=1;
    for (var it=1;it<=totalpage;++it){
        var itit="b"+it.toString();
        if(it>5)document.getElementById(itit).style.display="none";
        else document.getElementById(itit).style.display="";

    }
      });});
</script>';
```

Figure 2.11: First Button

Now we should set function for the buttons.In each button's function, we decide which divs should be shown and which divs should be hidden by the current page number. And we can change their display style by their div's id using the statement 'document.getElementById('b4').style.display='. Similarly, we also need to show or hide the buttons of each page. What should be mentioned is that when the total number of pages is less than 5, the function of pages' buttons' display should be write in different ways.

```
echo'
<script type="text/javascript">
 $(document).ready(function(){
$("#but3").click(function(){
   if(now>1){
    document.getElementById(now.toString()).style.display="none";
    document.getElementById((now-1).toString()).style.display="";
    now-=1;}else  document.getElementById("1").style.display="";

    for (var it=1;it<=totalpage;++it){
        var itit="b"+it.toString();
        if(it<(now-2)||it>(now+2))document.getElementById(itit).style.display="none";
        else document.getElementById(itit).style.display="";

    }if(totalpage>=1&&totalpage<=5)document.getElementById("b1").style.display="";
    else if(totalpage>=2&&totalpage<=5)document.getElementById("b2").style.display="";else if(
        totalpage>=3&&totalpage<=5)document.getElementById("b3").style.display="";else if(
        totalpage>=4&&totalpage<=5)document.getElementById("b4").style.display="";else if(
    totalpage==5)document.getElementById("b5").style.display="" ;else
    if(now>=1&&now<=2){document.getElementById("b4").style.display="";
     document.getElementById("b5").style.display="";}
     if(now>totalpage-2)document.getElementById("b"+(totalpage-4).toString()).style.display="";
     if(now>totalpage-1)document.getElementById("b"+(totalpage-3).toString()).style.display="";
       });});
</script>';
```

Figure 2.12: Previous Button

```
            echo'
<script type="text/javascript">
 $(document).ready(function(){
$("#but2").click(function(){
    if(now<totalpage){
    document.getElementById(now.toString()).style.display="none";
    document.getElementById((now+1).toString()).style.display="";
    now+=1;
       }
    else document.getElementById((totalpage).toString()).style.display="";
       for (var it=1;it<=totalpage;++it){
        var itit="b"+it.toString();
        if (document.getElementById(itit)){
        if(it<(now-2)||it>(now+2)) { document.getElementById(itit).style.display="none"; }
        else { document.getElementById(itit).style.display=""; }}

    }if(totalpage>=1&&totalpage<=5)document.getElementById("b1").style.display="";
    else if(totalpage>=2&&totalpage<=5)document.getElementById("b2").style.display="";else if(
        totalpage>=3&&totalpage<=5)document.getElementById("b3").style.display="";else if(
        totalpage>=4&&totalpage<=5)document.getElementById("b4").style.display="";else if(
    totalpage==5)document.getElementById("b5").style.display="" ;else

     if(now>=1&&now<=2){document.getElementById("b4").style.display="";
     document.getElementById("b5").style.display="";}
     if(now>totalpage-2)document.getElementById("b"+(totalpage-4).toString()).style.display="";
     if(now>totalpage-1)document.getElementById("b"+(totalpage-3).toString()).style.display="";
      });});
</script>';
```

Figure 2.13: Next Button

```
echo'
 <script type="text/javascript">
  $(document).ready(function(){
$("#but4").click(function(){
    document.getElementById(now.toString()).style.display="none";
    document.getElementById(totalpage.toString()).style.display="";
    now=totalpage;

    for (var it=1;it<=totalpage;++it){
        var itit=\'b\'+it.toString();
        if(it<totalpage-4)document.getElementById(itit).style.display="none";
        else document.getElementById(itit).style.display="";}
        });});
</script>';
```

Figure 2.14: Last Button

```
echo"

<script  type=\"text/javascript\">
 var oBtn=document.getElementsByClassName(\"button\");
 for (var it=6;it<=totalpage;++it){
        var itit='b'+it.toString();
        document.getElementById(itit).style.display=\"none\";


    }

for(var t=1;t<=totalpage;++t){
    oBtn[t+1].index=t;
    oBtn[t+1].onclick=function(){
       var pppp=(this.index).toString();
       document.getElementById(now.toString()).style.display=\"none\";
       document.getElementById(pppp).style.display=\"\";
       now=this.index;
       for (var it=1;it<=totalpage;++it){
        var itit='b'+it.toString();
        if(it<(now-2)||it>(now+2))document.getElementById(itit).style.display=\"none\";
        else document.getElementById(itit).style.display=\"\";

    }if(totalpage>=1&&totalpage<=5)document.getElementById('b1').style.display=\"\";
    else if(totalpage>=2&&totalpage<=5)document.getElementById('b2').style.display=\"\";else if(
        totalpage>=3&&totalpage<=5)document.getElementById('b3').style.display=\"\";else if(
        totalpage>=4&&totalpage<=5)document.getElementById('b4').style.display=\"\";else if(
    totalpage==5)document.getElementById('b5').style.display=\"\" ;else
    if(now>=1&&now<=2){document.getElementById('b4').style.display=\"\";
     document.getElementById('b5').style.display=\"\";}
     if(now>totalpage-2)document.getElementById('b'+(totalpage-4).toString()).style.display=\"\";
     if(now>totalpage-1)document.getElementById('b'+(totalpage-3).toString()).style.display=\"\";
    }}

</script>";
```

Figure 2.15: Button function

2.2.4 Demonstration

Some examples are demonstrated as follows.
Figure 2.16 shows the first page of an author.
Figure 2.17 shows the last page of a conference.

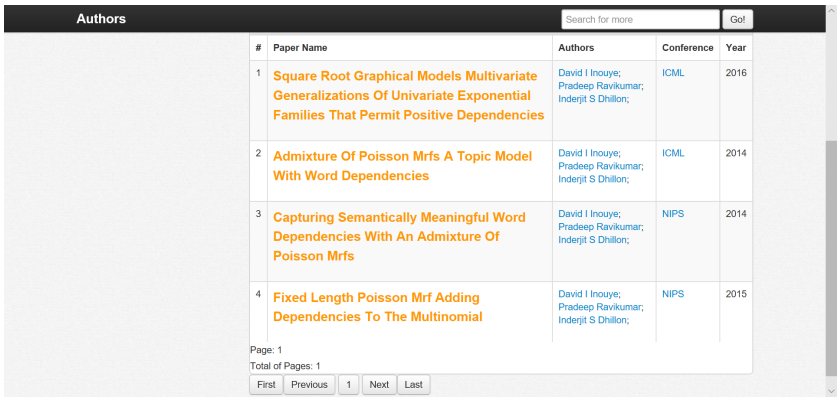


Figure 2.16: AUTHOR EXAMPLE

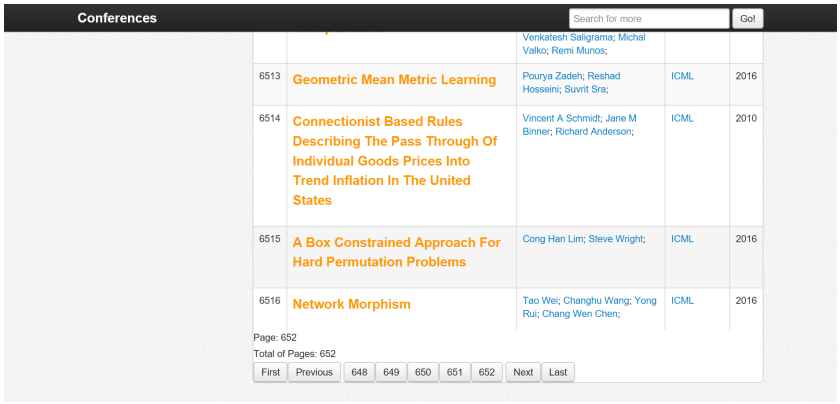


Figure 2.17: AUTHOR EXAMPLE

Chapter 3

Integrated Searching Bar

We use the copy field in solr to realize an integrated searching bar. By making copies of fields to one field, we can apply several distinct field types to a single piece of incoming information.

To construct a copy field, we first add a new field, for example a field 'keyword'. Then we choose "keyword" as destination and 'AuthorName', 'ConferenceName' and 'PaperName' as sources. Now, a query to 'keyword' can search all related information in three fields. In this way, we only need one integrated searching bar and do the query for one time to get all reults.

Chapter 4

Data Visualization

4.1 Statistical Graph

In our website, we have 12 charts made by echarts altogether to visualize the data in the form of 3D histogram, pie chart, line chart and bar chart.

4.1.1 3D histogram

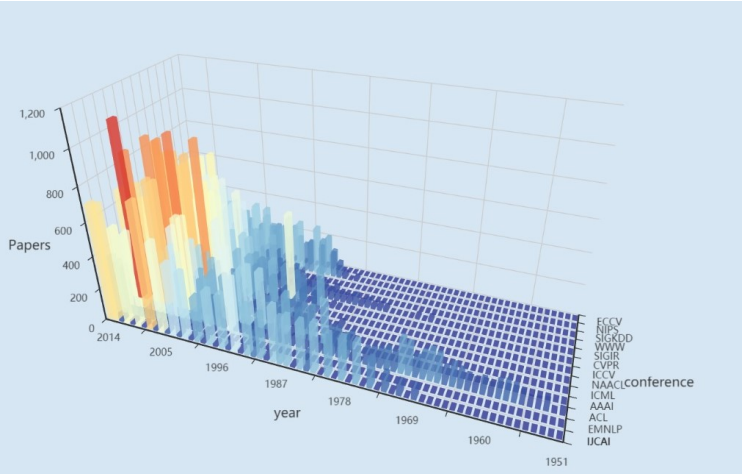


Figure 4.1: 3D chart in the homepage

In the homepage, we have a 3D chart to show the publication tendency of every conferences. This is a static chart that we prepare the data in advance. To construct a 3D chart in echarts, we need an array that concludes three-dimension coordinates. The z-axis value refers to the publication.

```python
import pymysql

conn = pymysql.connect(host='127.0.0.1',port=3306,user='root',passwd='',db='final',charset='utf8')
cursor = conn.cursor()

a = []

for i in range(13):
    for j in range(66):
        a.append([i,j,0])

conferences = ['ECCV','NIPS','SIGKDD','WWW','SIGIR','CVPR','ICCV','NAACL','ICML','AAAI','ACL','EMNLP','IJCAI']
confernece_ids = ['43001016','43319DD4','436976F3','43ABF249','43FD776C','45083D2F','45701BF3','45F914AD','465F7C62','46A05BB0','46DAB993','47167ADC','47C39427']

for i in range(13):
    conference_id = confernece_ids[i]
    sql = 'SELECT count(*), PaperPublishYear AS count FROM papers WHERE ConferenceID = %s GROUP BY PaperPublishYear ORDER BY PaperPublishYear'
    cursor.execute(sql,(conference_id))
    result = cursor.fetchall()
    for j in result:
        if (int(j[1]))>1950:
            year = int(j[1])
            for k in range(66*13):
                if a[k][0]==i and a[k][1]==year-1951:
                    a[k][2]=j[0]

print(a)
```

Figure 4.2: Get data form Mysql

4.1.2 Bar chart, pie chart and line chart

The codes are similar when we construct a bar chart, pie chart or line chart. We need two arrays to show the x-axis coordinates and their values. We first get the data from Mysql database and then convert the php array to javascript. SQL languages 'GROUP BY' and 'count(*)' are massively used to get the data.

The following three charts come from conference page, search page and author page respectively. All codes related to these charts are in file 'pages / XX / XX _charts.php'.

It's worth mentioning that to make all charts intelligible, we need to adjust the parameters in echarts option. For example, we set 'yAxis.minInterval: 1' so that decimal won't appear in y-axis labels. Also, we set 'xAxis.axisLabel.rotate: 40' so that all labels can all labels can be displayed(See Figure 4.4).

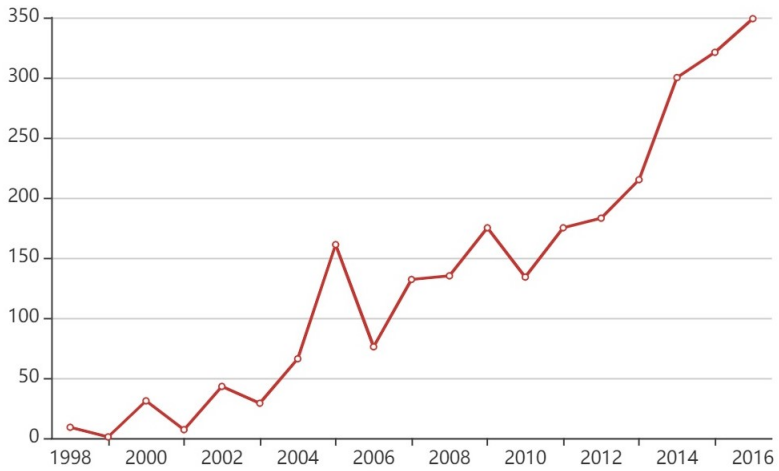


Figure 4.3: Line chart example

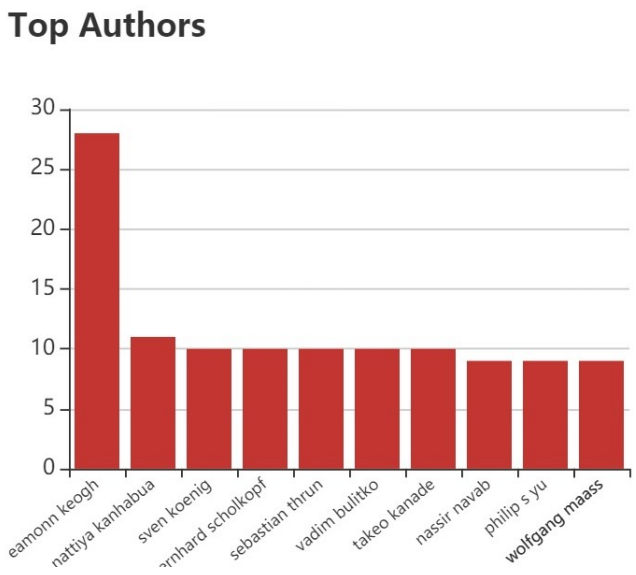


Figure 4.4: Bar chart example

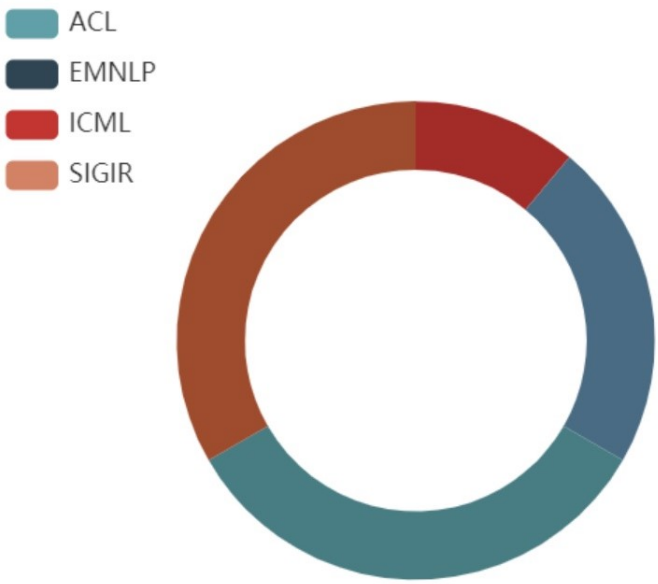


Figure 4.5: Pie chart example

4.2 Paper Relation Graph

A relation graph can visually reveal the relationship between papers, which we believe will be of great help to the users. For every paper, we've prepared a relation graph to show its reference papers, and the reference papers of its reference papers. Visitors can have a clear picture of how the paper is derived by looking at multiple levels of reference papers.

4.2.1 Description

We've applied a built-in template in echarts when making the relation graph. First we've fetched all the paper_reference relationship concerned, including multiple levels of reference relations, from the database. Then based on these data, we translate them into two arrays, namely nodes and links, which can be accepted by the echarts template. In the meantime, we add more information such as paper title, weight of the nodes and styles of the links into the array. Then the relation chart can be generated by echarts. In addition, we also wrote a jQuery function by ourselves in order to allow our users to visit the cited paper directly by clicking the nodes or the links. Some demo screenshots are listed below.

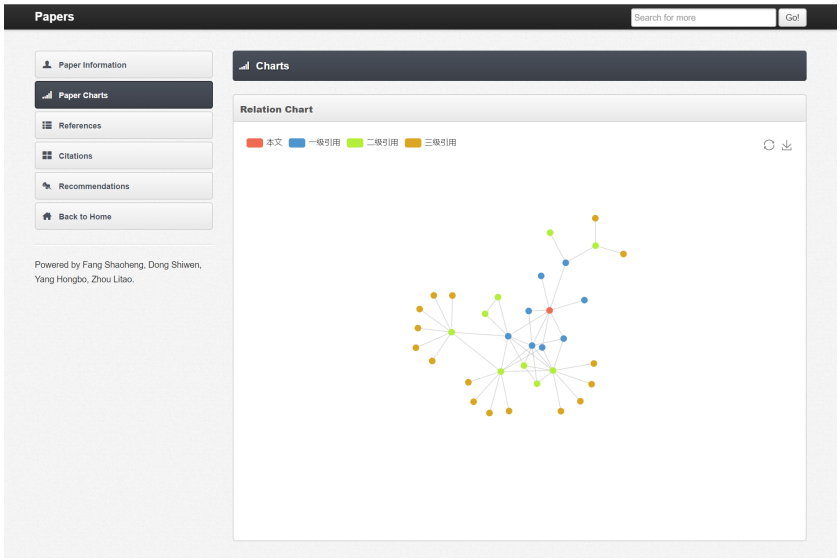


Figure 4.6: Relation Chart Page

4.2.2 Collect the Data

For convenience's sake, we first wrote a function in order to integrate the search of a paper's name into a function call. The ucwords() is a useful function

Figure 4.7: Multiple Display Modes & functions

in PHP, which can turn the first letter of every word into upper case. This can give our information display a better look.

```
function get_paper_name($link,$paper_id){
    $res = mysqli_query($link,"SELECT Title from papers where PaperID='$paper_id'");
    if ($res) {
            return ucwords(mysqli_fetch_array($res)['Title']);
        }
    else return ('Paper Not Found');
}
```

Figure 4.8: get_paper_name function

Then we use MySQL to get the direct reference of the paper. For these referenceIDs, we wrote a loop to add them to the nodes array, and put these first-level relations directly into the links array.

```
$links = [];
$nodes = [array('category'=>0,'name'=> $paper_id,'value'=>20, 'label'=> get_paper_name($link,$paper_id))];
$node_records = array($paper_id);


$connect = mysqli_fetch_all($result);
foreach ($connect as $connect_elem){
    $link_item = array('source'=>$connect_elem[1],  'target'=> $connect_elem[0] ,'name'=>'reference',
                       'label'=>get_paper_name($link,$connect_elem[1])."<br>is referenced by <br>".get_paper_name($link,$connect_elem[
                           0]));
    if (!(in_array($connect_elem[1],$node_records))){
        $node_item = array('category'=>1, 'name'=> $connect_elem[1], 'value'=>16, 'label'=> get_paper_name($link,$connect_elem[1]))
            ;
        array_push($node_records,$connect_elem[1]);
        array_push($nodes,$node_item);
    }
    array_push($links,$link_item);
}
```

Figure 4.9: First Level Data Collection

After that, we wrote a nested loop. The first level is to loop through the depth (i.e. the reference level), the second level is to loop through reference papers of the last reference item. The searching structure is just like building a tree. Note that we've created a helping array storing the paperids that we've already added into the nodes array, so that we can avoid overlapping nodes. Theoretically, the reference level (depth) can be streatched to infinity, but for practical use, we assume the depth of 4 will be fine. (See Figure 4.10 on next Page)

4.2.3 Set Echarts Options

The links and nodes arrays we've created above are just what the built-in echarts template requires. However, the initial graph configurations were not suitable for our information. (See Figure 4.11 on next Page) We need to block the display of node ids and loosen the distance between dinstinct nodes. After some manual regulating work, we make the graph look a lot better. Our major configuration codes are listed below. (See Figure 4.12 on next Page)

```php
$newconnect =array();
for ($depth=2;$depth<4;$depth+=1){
    foreach ($connect as $connect_elem){
        $result = mysqli_query($link, "SELECT paperID,referenceID FROM paper_reference2 WHERE paperID = '$connect_elem[1]'");
        $connection = mysqli_fetch_all($result);
        $newconnect = array_merge($newconnect,$connection);
        foreach ($connection as $connection_elem){
            $link_item = array('source'=>$connection_elem[1],  'target'=> $connection_elem[0],'name'=>'reference',
                               'label'=>get_paper_name($link,$connect_elem[1])."<br>is referenced by <br>".get_paper_name($link,$
                                   connect_elem[0]));
            if (!(in_array($connection_elem[1],$node_records))){
                $node_item = array('category'=>$depth, 'name'=> $connection_elem[1], 'value'=>(20-4*$depth), 'label'=>
                    get_paper_name($link,$connection_elem[1]));
                array_push($node_records,$connection_elem[1]);
                array_push($nodes,$node_item);
            }
            array_push($links,$link_item);
        }
    }
    $connect = $newconnect;
    $newconnect = array();
}

$nodes = json_encode($nodes);
$links = json_encode($links);
```

Figure 4.10: Higher-Level Data Collection

Figure 4.11: A Contrast between the Original and the Present Version of Paper Relation Charts

```
series : [
    {
        type:'graph',
        layout:'force',
        name : title,
        ribbonType: false,
        categories : categories,

        roam:true,
        focusNodeAdjacency:true,
        itemStyle: {
            normal: {
                label: {
                    show: false,
                    textStyle: {
                        color: '#333'
                    }
                },
        ......
        },
        force: {repulsion:80},
        useWorker: false,
        minRadius : 15,
        maxRadius : 25,
        gravity: 1.1,
        scaling: 1.1,
```

Figure 4.12: Echarts Options Configuration

4.2.4 Node Labels

The labels of our nodes should show the paper's name and the labels of our links should show the name of the two papers in the reference relation. This work has already been finished when we are collecting the data. We set the label of the nodes to be exactly what we want to show to our users. A further step we took is to add hyper-links to every nodes and edges. Note that the name of every node is its PaperID, which can be directly applied to the URL, while the name of every link is composed of the reference node and the origin node. We use the string carving function in PHP to get the first 8 character, representing the reference node, so users can access the reference page by clicking the links.

```
myChart.on('click', function (params) {
            var data=params.name.slice(0,8)

            if(data.source==undefined){
                nodeName=params.name
                window.open("./paper_info.php?paper_id="+data)
            }

});
```

Figure 4.13: Create Hyper-links for Node Labels

4.3 Big Charts using Gephi

As the saying goes, "One picture worths 1000 words.", a visualization picture is vital to show the connection between objects in some scales. To show an overall relationship, we choose **gephi** to visualize the data collected by MySQL. This method can help the part of E-charts to draw a better pictue in users' mind. There was a problem that troubled us for a long time. That is: how can we put the picture onto the pages. And it is finally solved by using package "ariutta/svg-pan-zoom " from GitHub. Here we'll give MySQL sequences and the way to draw a gephi picture(.svg).

4.3.1 MySQL Fetch

We choose to show the connection between authors in all 13 conferences. We established the connection by reference and citation. And this demands us to use 'join search' in MySQL. Example codes are given below.(We take ConferenceID= '47c39427' for an example.)

```
select i.AuthorName, j.AuthorName from(select h.AuthorID, g.AuthorName from(select e.ReferenceID, f.AuthorName
from (select d.AuthorID, c.ReferenceID from (select a.PaperID, b.ReferenceID from paper_reference2 b
inner join (SELECT papers.PaperID, papers.Title
FROM final.papers where papers.ConferenceID='47C39427') a on a.PaperID = b.PaperID) c
 INNER JOIN paper_author_affiliation d on c.PaperID=d.PaperID) e
 inner join authors f on f.AuthorID=e.AuthorID) g
 INNER JOIN paper_author_affiliation h on h.PaperID=g.ReferenceID) i
 inner join authors j on j.AuthorID=i.AuthorID
```

Figure 4.14: search example

Finally, remember to lead the data out in (.csv) format, in order that we can use the data in gephi easily.

4.3.2 Gephi Drawing

Gephi can use data in (.csv) format directly. After transmitting information into gephi, we can find a picture in the window. But this picture is so ugly and unclear that we cannot take it for visualization use derectly.

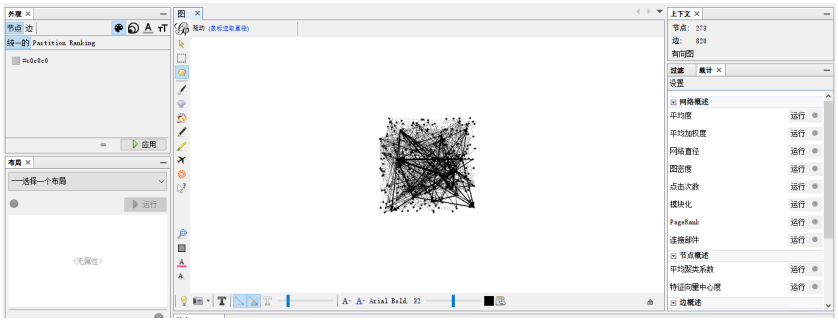


Figure 4.15: geghi: original picture

We may turn to gragh-drawing methods such as “force directed method” and “Fruchterman Reingold method”, to better display the characteristics of the relationship between authors.

Next, we can adjust the size of each node by degree, and give them different colors. It’s much easier for us to see the relation.

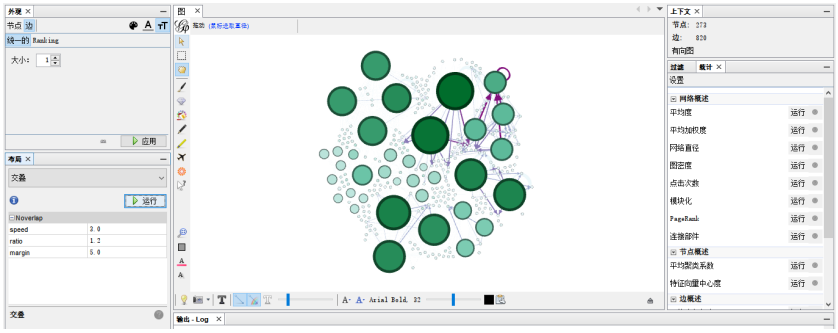


Figure 4.16: geghi: processed picture

Finally, we can transmit it as (.svg) file to load on the pages.

Chapter 5

Beautify the Pages

5.1 Index Beautification

There are altogether four parts in the index page: the top banner, the search box part, a chart and the bottom banner. The introduction of the chart is in the data visualization chapter.

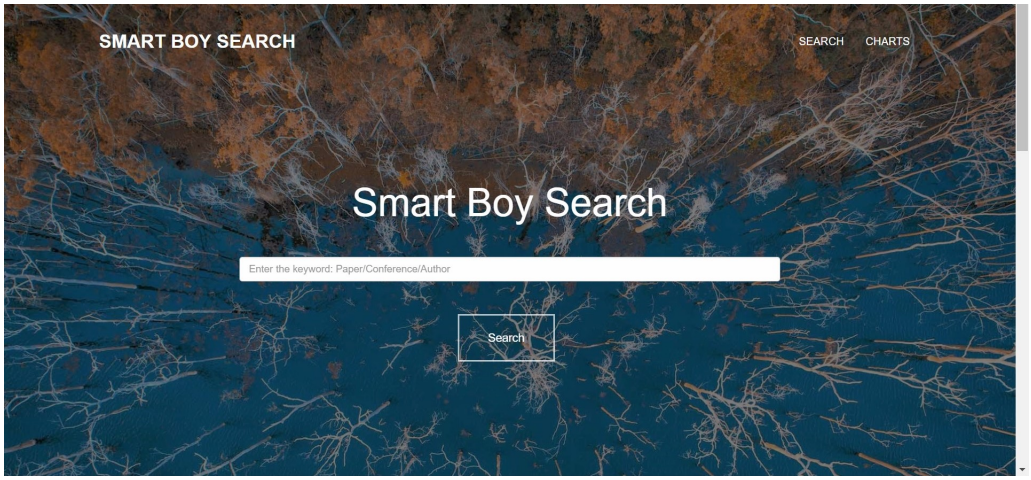


Figure 5.1: The index page

5.1.1 Top banner

In the top banner, we have our website's name 'Smart Boy' and two buttons, 'search' and 'chart'. By clicking the button, the page will jump to the search part or the chart part.

We use 'navbar' in bootstrap to complete out top banner. The links of 'search' and 'chart' are written in the nav tag(See in Figure 5.3).

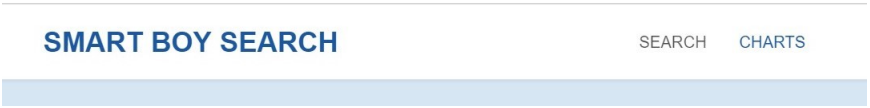


Figure 5.2: The top banner in index page

```
<nav class="navbar navbar-default probootstrap-navbar">
  <div class="container">
    <div class="navbar-header">
      <a class="navbar-brand" href="index.php" title="uiCookies:Frame">Smart Boy Search</a>
    </div>

    <div id="navbar-collapse" class="navbar-collapse collapse">
      <ul class="nav navbar-nav navbar-right">
        <li class="active"><a href="#" data-nav-section="home">Search</a></li>
        <li><a href="#" data-nav-section="charts">Charts</a></li>
      </ul>
    </div>
  </div>
</nav>
```

Figure 5.3: Codes of the top banner

5.1.2 Search box & main part

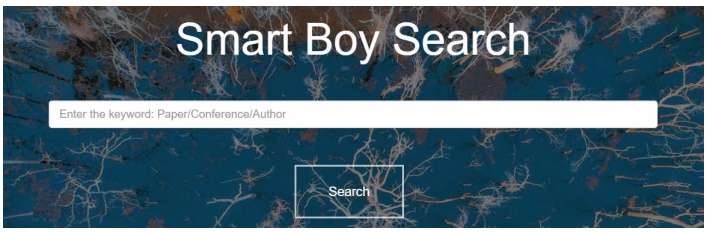


Figure 5.4: The search box in index page

```
<section class="probootstrap-hero prohttp://localhost/probootstrap/frame/#featuresbootstrap-slant" style="background-image: url(img/background1.jpg);" background-size=80%
data-section="home" data-stellar-background-ratio="0.7">
  <div class="container">
    <div class="row intro-text">
      <div class="col-md-8 col-md-offset-2 text-center">
        <h1 class="probootstrap-heading probootstrap-animate">Smart Boy Search</h1>
          <form class="form-horizontal" action="../../pages/search/search_info.php">
            <div class="form-group">
              <div class="col-md-12 text-center">
                <p class="probootstrap-animate"><input type="text" class="form-control" id="keyword" name="keyword" placeholder="Enter the keyword: Paper/Conference/Author"
                ></p>
                <input name='page' type='hidden' id='page' value=1 >
              </div>
            </div>
            <div class="form-group">
              <div class="probootstrap-subheading center">
                <p class="probootstrap-animate"><button type="submit" class="btn btn-default smoothscroll">Search</button></p>
              </div>
            </div>
          </form>

      </div>
    </div>
  </div>
</section>
```

Figure 5.5: Codes of the search box

First, we use bootstrap animation 'probootstrap-animate' to make all contents display gradually.

We also use 'placeholder' in the input tag to show sentence in the searching box

5.1.3 Bottom banner

At the bottom of the index page, we show some basic information of our website. We also have a link connected to our lesson website.

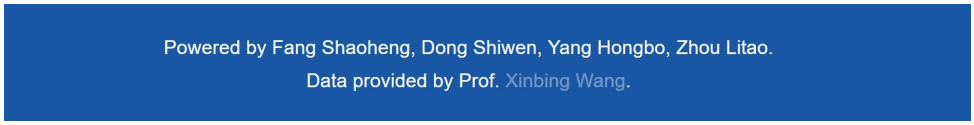


Figure 5.6: The bottom banner in index page

5.2 Pages Beautification

Our original page styles were derived from LAB3, which were very basic. With more and more functions implemented, we found that one page was not enough to carry all the information. As a result, we decided to split one page into several subpages carrying out different functions. For example, we split the paper.php page into information page, charts page, reference page, citation page and recommendation page. With the help of online resources, we fit our contents into a well-defined template. (tribute to http://www.cssmoban.com)

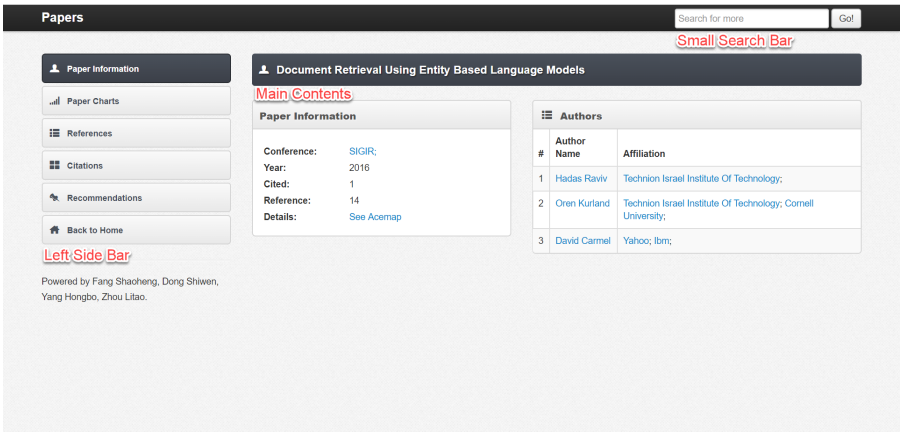


Figure 5.7: General Structure of the Pages

As the screenshot above reveals, all the pages share a similar layout - a top banner containing the page title and a small search bar, a right navigating column linked to all the related pages in this section and the right division where the main contents of the page are located.

5.2.1 Top Banner & Small Search Bar

Our template was originally designed for backstage admin system. In order to better fit our webpages, we replace the user login button with a small search bar. The search bar was a simple "form" application, but it turned out to be very useful when we are testing and using our pages. Codes of this part are listed as follows.

Figure 5.8: Top Banner

5.2.2 Left Navigation Bar

The template has already provided a series of button icons (derived from bootstrap framework) and a well-formatted list of buttons. To make our own navigation bar, we just need to change the icons according to the given names, and get the name and hyper-links right. Also, we can specify the class to be "active" in order to highlighten the button representing the current page.

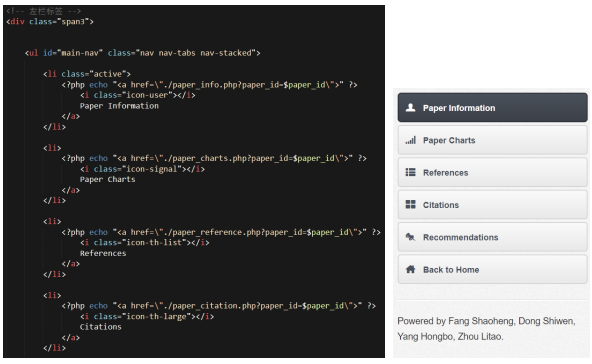


Figure 5.9: Left Navigation Bar

5.2.3 Main Contents

Since the template is based on the framework of Bootstrap, in the main contents, it follows that we should obey the rules of Bootstrap, such as containers, row classes, and the twelve column layout. Take a particular page for example, the generating process of the table through PHP is omitted. The layout for other pages are simlar in principle.

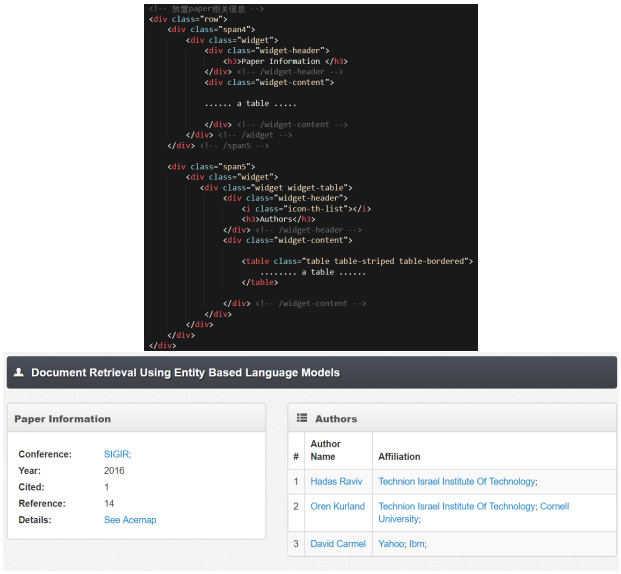


Figure 5.10: Main Contents

Chapter 6

MySQL Optimization in Affiliation Pages

The affiliation section is a section all made by ourselves. However, as has been briefly introduced in the Affiliation Pages section, new problems arise that the loading speed is too slow. This is because in our solution, the big paper_author_affiliation table will be searched everytime we display a new row in the webpage, which is clearly unnecessary. (See Figure 6.1)

```
for ($i=0;$i<$author_count;$i+=1){
    $author_id = $author_list[$i][0];
    $result = mysqli_query($link, "SELECT AuthorName from authors where AuthorID='$author_id'");
    $author_name = mysqli_fetch_array($result)['AuthorName'];
    echo "<tr>";
    echo "<td>";
    echo $i+1;
    echo "</td>";
    echo "<td>"."<a href=\"../authors/author_info.php?author_id=$author_id\">".ucwords($author_name)."</a></td>";
    $Affresult = mysqli_query($link, "SELECT Affiliations.AffiliationID, Affiliations.AffiliationName from (select
        AffiliationID, count(*) as cnt from paper_author_affiliation where AuthorID='$author_id' and AffiliationID
        null group by AffiliationID order by cnt desc) as tmp inner join Affiliations on tmp.AffiliationID = Affi
        AffiliationID");
    echo "<td>";
    if ($Affresult->num_rows!=0){
        foreach ($Affresult as $affline){
            $Affi_id = $affline['AffiliationID'];
            $Affi_name = ucwords($affline['AffiliationName']);
            echo "<a href=\"../affiliations/affiliation_info.php?affiliation_id=$Affi_id\">$Affi_name</a>;\n";}
        echo "</td>";
        }
    echo "<td>";
    $result = mysqli_query($link, "SELECT count(PaperID) from paper_author_affiliation where AuthorID='$author_id'
    $pub_count =  mysqli_fetch_array($result)[0];
    echo $pub_count;echo "</td>";
    echo "</tr>";
    }
echo "</tbody></table>";
echo "  </div> <!-- /widget-content -->";
```

Figure 6.1: Base Version in Affiliation Pages

6.1 Solution

Through practice, we found that the main factor dragging the loading time slow is the call by PHP for MySQL query. In order to reduce the loading time, we need to control the time we call a MySQL query. However, generally speaking, since we first need to get the authors based on the affiliation_id, and then find the detail information based on the author we have found in the first query, there is an unavoidable iterating structure in the process. The solution to this problem is not as easy as we assumed.

Eventually, we came up with the idea that all the searching work with MySQL can be integrated into one query. And we may just select the information we need from the total query results in order to display the information in the pages. This solution has restricted the searching query call to one single time. However, this will creat some difficulty when we want to echo the information onto the webpage.

6.2 Integrated MySQL Query in Authors List

Instead of searching the author information piece by piece, the searching query we write (See Figure 6.2) will return a group of rows showing the paper / affiliation information about one single author.

The basic idea about this integrated query is join the paper_author_affiliation table to itself, with restricting conditions of course. Our final result is a simplified paper_author_affiliation table, containing the author directly linked to the affiliaiton and the papers which might not be published in this affiliation, but whose authors once belonged to the affiliation. With the GROUP BY technique, the rows of every author can be returned group by group.

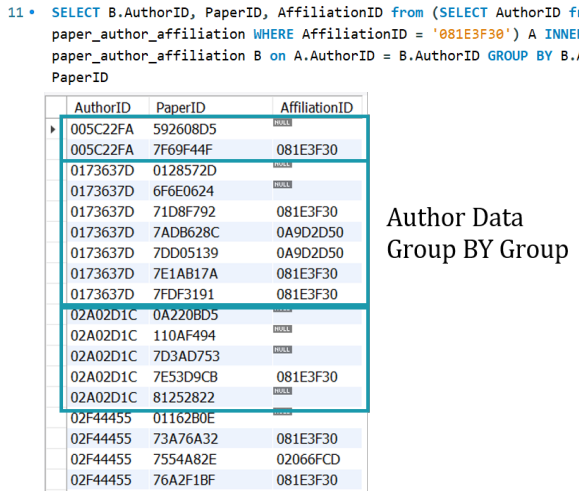


Figure 6.2: Integrated MySQL Query

6.3 Echoing Tricks in Authors List

Unlike what we did in the base version, where MySQL queries and PHP iterative loops are mixed together in an intuitive structure, here we are looping through the rows in the integrated searching results. (See Figure 6.2) So in every loop, we need to first judge whether this row is talking about the same author above or it starts a new one. Within the group corresponding to the same author, we just add one more affiliation into the webpage (before that we have to judge whether the affiliation is a new one, of course). While if the row marked the start of a new author, we have to end the previous row, begin a new row on the webpage. With these tricks, the final outlook on the webpage is just the same as the base version, but boasts a loading time within a second. The main part of the codes are listed below. (Initialization and Clean-up work omitted) The same tricks also apply to Paper List on the affiliation_paper page.

```
for ($i=1;$i<$result_rows;$i+=1){ // 从第二行开始
    if ($result[$i][0]!=$last_author) {$idx+=1; $last_author=$result[$i][0];$last_affiliation=
        array();// 另一位作者开始了
        echo "</td>";
        echo "<td>";echo $pub_count; echo "</td></tr>";$pub_count=1;//输出上一位作者的count
        // 新作者的信息
        echo "<tr><td>$idx</td><td>"."<a href=\"../authors/author_info.php?author_id=".$result[$i
            ][0]."\">".get_author_name($link,$result[$i][0])."</a></td>";
        echo "<td>";
        if ($result[$i][2]!=NULL) {
            array_push($last_affiliation,$result[$i][2]);
            echo "<a href=\"../affiliations/affiliation_info.php?affiliation_id=".$result[$i][2].
                "Affi_id\">".get_affiliation_name($link,$result[$i][2])."</a>;\n";}


    }
    else {
        if (!(in_array($result[$i][2],$last_affiliation)) AND $result[$i][2]!=NULL) {
        array_push($last_affiliation,$result[$i][2]);
        echo "<a href=\"../affiliations/affiliation_info.php?affiliation_id=".$result[$i][2]."
            Affi_id\">".get_affiliation_name($link,$result[$i][2])."</a>;\n";}
        $pub_count+=1;


        // 还是这位作者
```

Figure 6.3: Echoing Tricks

6.4 Integrated MySQL Query in Top Authors Charts

As has been discussed before in the Charts section, the top author charts require a list of author names and a list of numbers representing their publications counts. The same problem occurs that the data collecting work was taking too much time. We are also using MySQL joint search to replace PHP loops. We use count (DISTINCT) + GROUP BY commands to directly get the result. The MySQL queries are listed below. Further processing on the

searching results are fragile and hence omitted.

```
//top authors 需要数组authors1, author_num1, authors2, author_num2
// 通过一次查询，获得作者和作者的被引用数的表
$author_list2 = mysqli_fetch_all(mysqli_query($link,"SELECT AuthorID, count(distinct F.PaperID) FROM (SELECT B.AuthorID, PaperID
    from (SELECT AuthorID from paper_author_affiliation WHERE AffiliationID = '$affiliation_id') A INNER JOIN
    paper_author_affiliation B on A.AuthorID = B.AuthorID GROUP BY B.AuthorID, PaperID) E INNER JOIN paper_reference2 F on E.
    PaperID = F.ReferenceID GROUP BY AuthorID"));
$author_list1 = mysqli_fetch_all(mysqli_query($link,"SELECT AuthorID, count(distinct PaperID) FROM (SELECT B.AuthorID, PaperID
    from (SELECT AuthorID from paper_author_affiliation WHERE AffiliationID = '$affiliation_id') A INNER JOIN
    paper_author_affiliation B on A.AuthorID = B.AuthorID GROUP BY B.AuthorID, PaperID) G GROUP BY AuthorID"));
```

Figure 6.4: Integrated MySQL Query in Top Authors Charts

All the three parts introduced above make up our optimization of the affiliation section.

Appendix

Division of Writing

Chapter	Section	Writer
Preface		*Litao Zhou*
I. Enrich Contents	Paper & Conference Pages	*Hongbo Yang*
	Affiliation Pages	*Litao Zhou*
II. Leaf Turning	Version I: by PHP parameters	*Shiwen Dong*
	Version II: by jQuery	*Shiwen Dong*
III. Integrated Search Bar		*Shaoheng Fang*
IV. Visualization	Statistical Graph	*Shaoheng Fang*
	Paper Relation Graph	*Litao Zhou*
	Big Charts using Gephi	*Hongbo Yang*
V. Beautify Pages	Index Beautification	*Shaoheng Fang*
	Pages Beautification	*Litao Zhou*
VI. MySQL Optimization in Affiliation Pages		*Litao Zhou*
Conclusion		*Litao Zhou*

Acknowledgements

Contents Page Templates from http://www.cssmoban.com
SVG-pan-zoom Plug-in from https://github.com/ariutta/svg-pan-zoom
Index Page Templates from ...?

Epilogue

After a semester's study and a month's efforts on this project, we've worked out a well-formatted, information-rich, and smooth-running website on academic papers, as has been shown at full length in this report. Although our current website leaves much to be desired and more functions to be expected, due to the limitation of time, our developing story have to mark an end here. As the saying of Shakespeare goes, 'What's past, it's prologue.' The insightful, inspiring and illuminating course has become the past, so has the website project we've worked on during the past unforgettable days. However, our gratefulness to the instruction from our professor and teaching assistants and our appreciation of the inspirations from our excellent classmates will not be erased. Neither will our interest and passion in practicing and learning web-building fade away. These, in particular, will become the prologue of our future goals.