Web Search & Mining

—— Option A

钱爱娟 119033910115
徐艺 119033910122
李依安 119033910048

June 15, 2020

Contents

1 Introduction

1.1 Aims and Objectives

In this project, we will build a simple Chinese WestLaw system based on the court records of legal cases in China in Chinese provided.

1.2 Data description

We utilize three types of data in the project, including *data1*, *data2*, and *instruments*. In boolean search and fuzzy search, we use *data1* and *data2*, while in the query type we use *instruments*. The fields used for each type of data are shown in Table 1, Table 2, and Table 3, respectively. Besides, the specific meaning of each field in different data is listed in appendix A.

Since the data is contained in a large number of files, for the convenience of reading and operation, we store the data in the database after preprocessing. Each type of data corresponds to one table, i.e., a total of three tables.

From Table 1 and Table 2, we find that there is the same field *'caseCode'*. We perform union operation on the two tables against this field, and find that only a small part of the value of field *'caseCode'* is the same relative to the total amount of data. Therefore, in order to avoid a large number of missing values in the merged table, we do not merge *data1* and *data2* at last.

id	iname	caseCode	age	sexy	cardNum	bussinessEntity
courtName	areaName	gistId	regDate	gistUnit	duty	performance
performedPart	unperformPart	disruptTypeName	publishDate	qysler		

Table 1: *Fields of data1 used in this project.*

caseCode	iname	iaddress	imoney	ename	courtName & phone

Table 2: *Fields of data2 used in this project.*

id	caseCode	title	type	cause	department	level	closingDate	content

Table 3: *Fields of instruments used in this project.*

1.3 Inverted index

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance. The major steps include (1) Tokenization. (2) Construct the term dictionary and the posting list.

1.3.1 Tokenization

Tokenization is to tokenize each document and find terms to be indexed. The main point here is data processing. Because each document consists of some attributes, the tokenization process for a document is to tokenize each attribute of it. The detailed process is:

1. Remove useless punctuation and spaces;
2. Remove stop words;
3. Use package *jieba* to segment each attribute.

After that, we turn each document into a list of tokens. Especially, we do not tokenize all attributes and ignore the "id" field. Besides, for improving the search result , we do not segment and retain some attributes which are short and often to search such as "caseCode".

1.3.2 Construct term dictionary and posting list

Having got all tokens to be indexed for a document, we can construct the term dictionary and posting list. We assign a DocID for each document and each DocID occupy 4 bytes. For the term dictionary, each case includes a term as the key and a tuple as the value:

$$\{\text{term: (document frequency, offset}\}$$

The offset is the beginning point in the posting list file. The posting list file will save posting lists of all terms continuously. To find the posting list of a term, we can first get its document frequency and offset in the dictionary. Then its posing list starts at offset and covers (document frequency*4) in length.

Finally, we build a dictionary *dictionary* and postings *postings* for *data1* and *data2*, and a dictionary *ins_dictionary* and postings *ins_postings* for *instruments*.

1.4 System overview

1.4.1 System framework

According to the requirements of the project and the data provided, we propose $CWLsystem$, the whole framework and processing flow of our system are shown in Figure 1. After a series of processing of the raw data, the dictionaries and postings will be generated. When users perform the search on the web page, the back-end search algorithms will return the results based on the dictionaries and postings, and then display the results on the web page. When users rank the data, the web page will also display the sorted data according to the results given by the ranking algorithm.

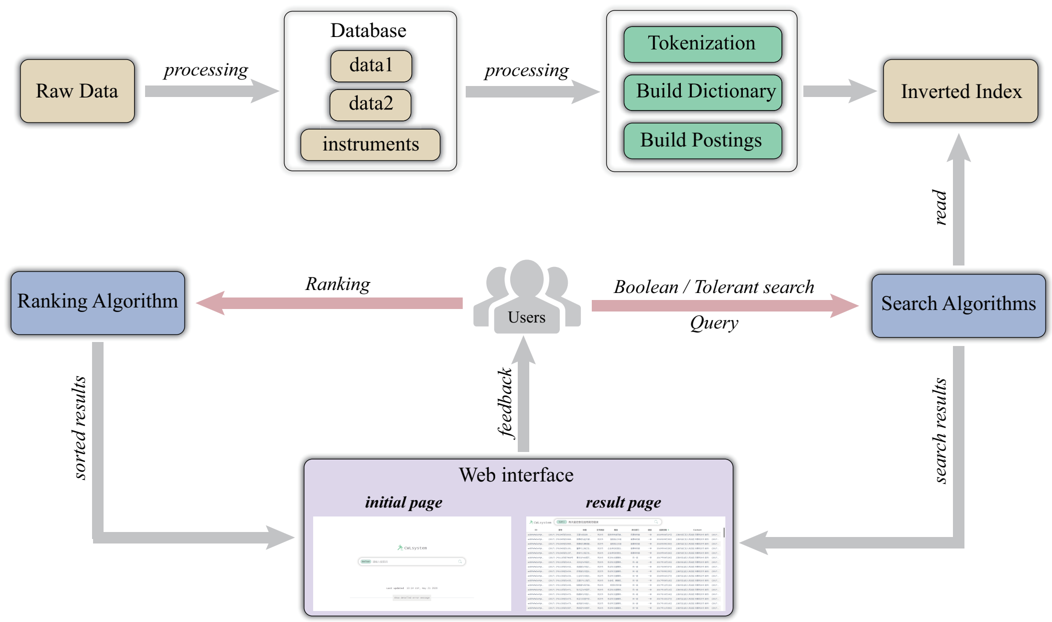

Figure 1: *The framework of our system.*

1.4.2 System functions

Our system *CWLsystem* supports the following operations:

- **Boolean search:** Based on the search keys provided by the user and the operations between the keys, including AND, OR, NOT, all the relevant documents or data will be returned to the user.

- **Tolerant (fuzzy) search:** While searching data with attributes including person name, city, case code, company name, etc., users may wrongly type some information. And our system will support tolerant search to solve the problem of no exact match between the search term and document term. Moreover, the fuzzy results will be ranked according to the relevance to users' search keys.

- **Query:** When users search for legal instruments with a query sentence, our system will return the instruments according to the relevance to the query sentence.

- **Ranking:** For different search types, our system can sort the returned data or documents by several fields.

1.4.3 User guide

1. The initial page of our system is shown in Figure 2. The user can click the green button to select the search type, type the keys or query to be searched in the input box, and then press the *"Enter"* to perform the search.

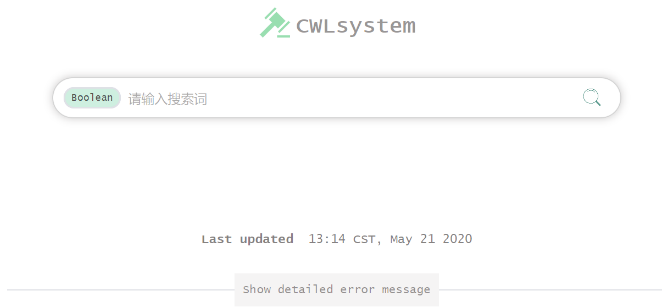

Figure 2: *The initial page of the CWLsystem.*

2. The results page is shown in Figure 3. The search results are displayed in the form of a table, where each row represents a document, and each column represents a field.

ID	案号	标题	文书类别	案由	承办部门	级别	结案日期	Content
adGFoPaOoMjA...	（2019）沪民终76号	上诉人上海琪峰...	裁定书		海事及海商审判庭	二审	2019年09月11日	上海市高级人民法院 民事裁定书 案号：（2019）...
adGFoPaOoMjA...	（2019）沪民申1067号	上海星琴文化传...	裁定书		刑事审判庭（未...		2019年09月10日	上海市高级人民法院 民事裁定书 案号：（2019）...
adGFoPaOoMjA...	（2019）沪民申1070号	茅耘与上海嘉隆...	裁定书	物件损害责任纠纷	刑事审判庭（未...	申诉	2019年09月06日	上海市高级人民法院 民事裁定书 案号：（2019）...
adGFoPaOoMjA...	（2019）沪民终288号	吴华与上海建工...	判决书	建设工程施工合...	民事审判庭（环...	二审	2019年09月06日	上海市高级人民法院 民事判决书 案号：（2019）...
adGFoPaOoMjA...	（2019）沪0115民初5710...	曹某某与戴某某...	判决书		外高桥法庭	一审	2019年09月04日	上海市浦东新区人民法院 民事判决书 案号：（20...
adGFoPaOoMjA...	（2019）沪0101民初1309...	王继泉与上海融...	裁定书	借款合同纠纷	商事审判庭	一审	2019年09月03日	上海市黄浦区人民法院 民事裁定书 案号：（2019...
adGFoPaOoMjA...	（2019）沪03民终93号	苏州市第六建筑...	判决书	铁路修建合同纠纷	民事审判庭（环...	二审	2019年09月02日	上海市第三中级人民法院 民事判决书 案号：（20...
adGFoPaOoMjA...	（2019）沪民申1111号	上海缘保投资中...	裁定书	合伙协议纠纷	商事审判庭（破...	申诉	2019年08月27日	上海市高级人民法院 民事裁定书 案号：（2019）...
adGFoPaOoMjA...	（2019）沪74民终509号	曹国明与平安银...	判决书	金融借款合同纠纷	综合审判二庭	二审	2019年08月26日	上海金融法院 民事判决书 案号：（2019）沪74...
adGFoPaOoMjA...	（2019）沪74民终560号	李彦斌与平安银...	判决书	信用卡纠纷	综合审判一庭	二审	2019年08月23日	上海金融法院 民事判决书 案号：（2019）沪74...

Figure 3: *The result page of the CWLsystem.*

3. Users can click on each row of the table to view the detailed information of the corresponding document. The detailed information page is shown in Figure 4.

字段	内容
ID	adGFoPaOoMjAxOaOpu6YwMTE3w/Gz9TExNTgzusUmd3N4aD0xz
案号	（2019）沪0117民初11583号
标题	上海景时实业有限公司与扬州新彩服饰工艺辅助材料有限公司买卖合同纠纷一审民事裁定书
文书类别	裁定书
案由	买卖合同纠纷
承办部门	商事审判庭
级别	一审
结案日期	2019年08月05日
Content	上海市松江区人民法院 民事裁定书 案号：（2019）沪0117民初11583号 原告：上海景时实业有限公司，住所地上海市松江区。 法定代表人：徐丽萍，总经理。 委托诉讼代理人：黄骅，上海钟颖律师事务所律师。 被告：扬州新彩服饰工艺辅助材料有限公司，住所地江苏省扬州市。 法定代表人：郑金梅，职务不详。 原告上海景时实业有限公司与被告扬州新彩服饰工艺辅助材料有限公司买卖合同纠纷一案，本院于2019年5月27日立案，依法适用小额诉讼程序。现原告以被告已还清拖欠的货款为由向本院申请撤回起诉。 本院认为，当事人可在法律规定的范围内处分自己的民事权利和诉讼权利。原告上海景时实业有限公司向本院申请撤回起诉，并无不当，应予准许。依照《中华人民共和国民事诉讼法》第一百四十五条第一款之规定，裁定如下： 准许原告上海景时实业有限公司撤回起诉。 本案适用小额诉讼程序，案件受理费予以免收。 审判员 陈巨澜 书记员 彭欣怡 二〇一九年八月五日

Figure 4: *The detailed information page of the CWLsystem.*

4. After this search, users can stay on the result page or return to the initial page to perform the next search.

2 Boolean search

The three main Boolean operators are AND, OR and NOT. In this section, we introduce the Boolean search algorithm and explain how to retrieve all the relevant documents for a query that contains search keys and operations between keys with the help of posting lists and the Boolean retrieval model. To avoid confusion, we treat the Chinese brackets (’（’ and ’）’) in query as part of a term and the English brackets (’(’ and ’)’) as part of a Boolean operator.

2.1 Methodology

The Boolean retrieval process can be described as the following three steps: 1) transform the search string (including operations and search keys) into a suffix expression; 2) locate each query term in the dictionary and retrieve its postings; 3) combine the posting lists according to the operations. Each operation is a binary operation, namely, we process two query terms at a time, store the results of the current operation and then proceed to the next one.

2.2 Query Preprocessing

Since the input query is an infix expression containing multiple operations with different priorities, we need to preprocess the query to convert it into a suffix expression. Algorithm 1 illustrates the detailed procedure.

Algorithm 1 Query Preprocessing

Input: *infix_tokens*: the input query
Output: *output*: the suffix output

```
output = []
precedence = {'NOT': 3, 'AND': 2, 'OR': 1, '(': 0, ')': 0}
for token in infix_tokens do
    if token == '(' then
        operator_stack.append(token)
    else if token == ')' then
        pop all operators from operator_stack onto output until we hit left bracket
    else if token in precedence then
        pop operators from operator_stack to queue if they are of higher precedence
    else
        output.append(token)
while operator_stack do
    output.append(operator_stack.pop())
return output
```

2.2.1 NOT

The purpose of the Boolean NOT operator is to obtain documents that do not contain the current term. In practice, we return the list of *docIDs* which is the compliment of the posting list of the given term.

2.2.2 OR

The goal of the OR operation is to efficiently combine posting lists so as to be able to quickly find documents that contain either term. Algorithm 2 shows the simple merge algorithm of two posting lists with a logical OR operation.

We maintain pointers into both lists and walk through the two posting lists simultaneously, in time linear in the total number of posting entries. At each step, we compare the *docID* pointed to by both pointers. If they are the same, we put that *docID* in the results list, and advance both pointers. Otherwise we put the smaller *docID* in the result list and advance its pointer. If the lengths of the posting lists are x and y, the intersection takes $O(x+y)$ operations.

Algorithm 2 Boolean OR

Input: $List_1$: docID list on the left; $List_2$: docID list on the right
Output: $results$: results of the Boolean OR operation

$p_1 = 0$
$p_2 = 0$
$results = []$
while $p_1 < len(List_1)$ or $p_2 < len(List_2)$ **do**
    if $p_1 < len(List_1)$ and $p_2 < len(List_2)$ **then**
        if $List_1[p_1] == List_2[p_2]$ **then**
            $results.append(List_1[p_1])$
            $p_1 += 1$
            $p_2 += 1$
        else if $List_1[p_1] < List_2[p_2]$ **then**
            $results.append(List_1[p_1])$
            $p_1 += 1$
        else
            $results.append(List_2[p_2])$
            $p_2 += 1$
    else if $p_1 \geq len(List_1)$ **then**
        $results.append(List_1[p_1])$
        $p_1 += 1$
    else
        $results.append(List_2[p_2])$
        $p_2 += 1$
return $results$

2.2.3 AND

The AND operation is similar to the OR operation, except that the goal of AND is to find documents that contain both terms by intersecting two posting lists. Therefore, the intersection also takes $O(x+y)$ operations. However, we can accelerate this intersection process using skip pointers to skip certain postings that will not figure in the search results. Algorithm 3 illustrates the merge algorithm for the AND operation with skip pointers. Recall that in the previous algorithm, we advance both pointers by one step at a time. In algorithm 3, however, we will first check the skip pointer lists to see if we can move further.

Algorithm 3 Boolean AND

Input: $List_1$: docID list on the left; $List_2$: docID list on the right
Output: $results$: results of the Boolean AND operation

```
p1, p2 = 0, 0
results = []
while p1 < len(List1) and p2 < len(List2) do
    if List1[p1] == List2[p2] then
        results.append(List1[p1])
        p1 += 1
        p2 += 1
    else if List1[p1] < List2[p2] then
        if hasSkip(p1) and List1[Skip(p1)] <= List2[p2] then
            p1 = Skip(p1)
        else
            p1 += 1
    else
        if hasSkip(p2) and List2[Skip(p2)] <= List1[p1] then
            p2 = Skip(p2)
        else
            p2 += 1
return results
```

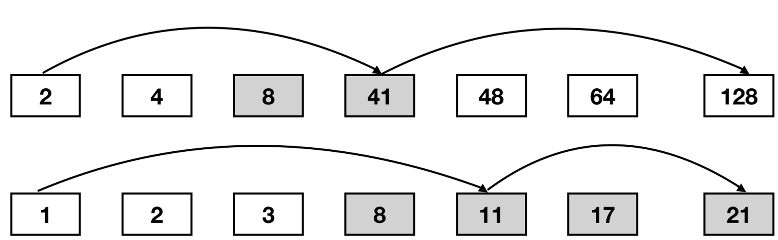

Figure 5: *Example of skip pointers.*

Figure 5 shows an example. Suppose we currently find a match at 8. Then we advance both pointers to get 41 and 11. We also notice that the next *docID* in the skip list is 21, which is also smaller than 41. In such situation, we can continue increasing the pointer until there is no skip pointers or the next *docID* is greater than 41.

2.3 Result

In this section, we will show three examples of boolean search. Because there are too many results, only the first few results and related fields are displayed. Besides, the keys in the search are marked with green strokes.

- Search **"张灿荣 OR 酒泉市"**, the result is shown in Figure 9.

ID	姓名	执行法院	执行依据单位	义务
100000090	张灿荣	惠安县人民法院	惠安县人民法院	被告张灿荣应于本判决生效之日起...
100000178	刘红兵	酒泉市肃州区人民法院	甘肃省酒泉市肃州区法院	支付各项赔偿共计84111.83元
101144690	杨建军	酒泉市肃州区人民法院	酒泉市肃州区人民法院	被告杨建军返还原告酒泉市肃州区...
101744710	李志银	瓜州县人民法院	瓜州县人民法院、酒泉市...	被告李志银给付原告刘彦顺房屋修...
105189954	郑洪山	酒泉市肃州区人民法院	酒泉市肃州区人民法院	被告支付原告170827元
105978795	张灿荣	惠安县人民法院	惠安县人民法院	被告张灿荣应于本判决生效之日起...
106201133	王鑫	酒泉市肃州区人民法院	酒泉市中级人民法院	被告支付原告248254元

data1

被执行人地址
甘肃省酒泉市肃州区三墩镇二墩村5组新35号
甘肃省酒泉市肃州区东环东路31号121室
甘肃省酒泉市金塔县中东镇王子庄村一组49号
甘肃省酒泉市肃州区南苑小区17号楼122号
甘肃省酒泉市肃州区北环西路新世纪4号楼3-3-2
甘肃省酒泉市肃州区北环西路新世纪4号楼3-3-2
甘肃省酒泉市肃州区南大街顺河家园3单元601室

data2

Figure 6: *The first example "张灿荣 OR 酒泉市".*

- Search **"王雪英 AND 惠安县"**, the result is shown in Figure 7.

ID	姓名	执行法院	省份	执行依据单位	义务
100000123	王雪英	惠安县人民法院	福建	惠安县人民法院	一、被告福建立恒涂料有限公司、王雪英结欠原告杨...
101393367	王雪英	惠安县人民法院	福建	惠安法院	支付申请人1900000元
516596644	王雪英	惠安县人民法院	福建	惠安县人民法院	
701419149	王雪英	惠安县人民法院	福建	惠安县人民法院	一、被告王雪英应于本判决生效之日起十日内偿还原...
702503764	王雪英	惠安县人民法院	福建	泉州市中级人民法院	一、被告福建立恒涂料有限公司、王雪英确认尚欠原...
702503767	王雪英	惠安县人民法院	福建	泉州市中级人民法院	一、被告王雪英应于本判决生效之日起十日内偿还给...
703423084	王雪英	惠安县人民法院	福建	泉州市中级人民法院	一、被告福建省惠建发建设工程有限公司应偿还原告...

data1

Figure 7: *The second example "王雪英 AND 惠安县".*

- Search **"张灿荣 OR (王雪英 AND 惠安县)"**, the result is shown in Figure 8.

ID	姓名	执行法院	省份	执行依据单位	义务
100000090	张灿荣	惠安县人民法院	福建	惠安县人民法院	被告张灿荣应于本判决生效之日起十日内偿还原告刘煌波...
100000123	王雪英	惠安县人民法院	福建	惠安县人民法院	一、被告福建立恒涂料有限公司、王雪英结欠原告杨明珠...
101393367	王雪英	惠安县人民法院	福建	惠安法院	支付申请人1900000元
105978795	张灿荣	惠安县人民法院	福建	惠安县人民法院	被告张灿荣应于本判决生效之日起五日内偿还原告吴灿辉...
516596644	王雪英	惠安县人民法院	福建	惠安县人民法院	
700353729	张灿荣	郁南县人民法院	广东		(2015) 云郁法刑初字第142号
700965291	张灿荣	靖江市人民法院	江苏	江苏省靖江市人民法院	一、被告张灿荣归还原告中国银行股份有限公司靖江支行...

data1

Figure 8: *The third example* “张灿荣 *OR* (王雪英 *AND* 惠安县)” .

3 Tolerant (fuzzy) search

Tolerant (fuzzy) search is to tolerate type mistakes for those search keys in boolean search. The main point for tolerant search is to find similar terms with these search keys and to generate or rank final documents based on these groups of terms.

3.1 Methodology

Our methodology is divided into three parts: (1) For a search key, find all similar terms and their corresponding documents. (2) Calculate final scores for all related documents as boolean search goes. (3) Rank these documents. Here data1 and data2 are needed and we use the *dictionary* and *posting_lists*. The implementation details are as follows:

3.1.1 Generate related documents and their initial scores

For a search key, we use Levenshtein distance to calculate the edit distance of the search key and all terms in *dictionary* so that we can find similar ones. Assume string s_1 with length of m and s_2 with length of n, we can apply dynamic programming as follows:

$$d_{[i,j]} = \begin{cases} 0 & i = 0 \text{or} j = 0 \\ \min\left(d_{[i-1,j]} + 1, d_{[i,j-1]} + 1, d_{[i-1,j-1]}\right) & x_i = y_j \\ \min\left(d_{[i-1,j]} + 1, d_{[i,j-1]} + 1, d_{[i-1,j-1]} + 1\right) & x_i \neq y_j \end{cases}$$

where $d_{[i,j]}$ is the edit distance of s_1 with length of i and s_2 with length of j. $d_{[i-1,j]} + 1$ represents inserting a character in s_2. $d_{[i,j-1]} + 1$ represents removing a character in s_1. $d_{[i-1,j-1]} + 1$ means replacing a character. For initialization, we need a matrix of $(m+1) \times (n+1)$, and $d_{[i,0]} = i$, $d_{[0,j]} = j$.

Then for a search key k and a term t, we note the distance between them as $D(k,t)$. The similar score for k and t is:

$$score(k,t) = \frac{len(k) + len(t) - D(k,t)}{len(k) + len(t)}$$

For all terms t, if $score(k,t) > Threshold$ (Here we set Threshold as 0.7), we think the term t is similar to search key k. In this way, we can find all similar terms $T(key)$ in *dictionary* and the related documents (noted as $Doc(t)$) in *posting_lists* according to a similar term t. For all term t in $T(key)$, we can generate the union sets D of all related documents $Doc(t)$. The initial score of each documents d in D is the maximum $score(k,t)$ among all related terms t, where t has d in its posting list.

In conclusion, for a search key k, we can find all related documents and their initial scores like $[(d_1, s_1), (d_2, s_2), ...]$.

Algorithm 4 Generate related documents and their final scores

Input: a suffix expression list E, *dictionary*, *posting_lists*
Output: (related document, final score) list: $[(d_1, s_1), (d_2, s_2), ...]$

$results\_stack = []$
while E **do**
  $token = E.pop()$
  find all related documents and their initial scores $[(d_1, s_1), (d_2, s_2), ...]$ for $token$
  $result = [(d_1, s_1), (d_2, s_2), ...]$
  sort result according to docID
  if $token$ is not operations **then**
    find all related terms list $T(token)$ in *dictionary*
  else if $token = AND$ **then**
    $right\_operand = results\_stack.pop()$
    $left\_operand = results\_stack.pop()$
    $result = boolean\_AND'(left\_operand, right\_operand)$
  else if $token = OR$ **then**
    $right\_operand = results\_stack.pop()$
    $left\_operand = results\_stack.pop()$
    $result = boolean\_OR'(left\_operand, right\_operand)$
  else if $token = NOT$ **then**
    $right\_operand = results\_stack.pop()$
    $result = boolean\_NOT'(right\_operand)$
  $results\_stack.append(result)$
return $results\_stack$

3.1.2 Generate final scores

In boolean search, we have already transformed the search string (including operations and search keys) into a suffix expression (noted as E). As analysing the expression, we can collect all related documents and give each one a score. The implementation algorithm is in Algorithm 4.

The function $boolean\_AND'$, $boolean\_OR'$, $boolean\_NOT'$ here are to combine the tuple list $[(d_1, s_1), (d_2, s_2), ...]$ and update the document score. The updating rule is defined as:

1. $boolean\_AND'$: for a interacted doc d, its score is the minimum value in both sets. "AND" means doc d should satisfy both requirements, so we strictly limited the score of interacted documents.

2. $boolean\_OR'$: for a union doc d, its score is the maximum value in both sets. "OR" means doc d just should satisfy only one requirement, so we loose the score of union documents.

3. $boolean\_NOT'$: we won't update anything here, because "NOT" will filter all documents we do not need.

3.1.3 Rank Documents

In Section 3.1.2, we have generated all related documents and their final scores for a bool search string, so we just need to rank these documents in descending order according to their scores now.

3.2 Result

In this section, we will show two examples of tolerant search. Because there are too many results, only the first few results and related fields are displayed.

- Search **"上海徐汇人民法院"**, and the result is shown in Figure 9.

性别	身份证	法人	执行法院	省份	执行依据文号	立案时间	执行依据单位
男	3101...		上海市徐汇区人民法院	上海	580	2015年02月04日	上海市徐汇区人民法院
男	6101...		上海市徐汇区人民法院	上海	580	2014年12月17日	上海市徐汇区人民法院
男	3208...		上海市徐汇区人民法院	上海	580	2014年11月25日	上海市徐汇区人民法院
男	3702...		上海市徐汇区人民法院	上海	580	2014年07月11日	上海市徐汇区人民法院
女	3101...		上海市徐汇区人民法院	上海	580	2014年07月11日	上海市徐汇区人民法院
男	3306...		上海市徐汇区人民法院	上海	580	2015年03月23日	上海市徐汇区人民法院
男	3101...		上海市徐汇区人民法院	上海	580	2015年04月13日	上海市徐汇区人民法院
女	1101...		上海市徐汇区人民法院	上海	580	2014年07月26日	上海市徐汇区人民法院
男	3501...		上海市徐汇区人民法院	上海	580	2014年11月26日	上海市徐汇区人民法院

Figure 9: *The first example of tolerant search.*

- Search **“(2017) 沪 0112 执 5984 号”**, and the result is shown in Figure 10.

ID	姓名	案号	年龄	性别	身份证
51669...	沈阳...	(2017) 辽0112执594号	0		75552075X
51715...	杨秀华	(2017) 鲁0112执598号	40	女	3709231976*...
54334...	李峰	(2017) 云0112执1584号	43	男	5301281972*...
54334...	李贵珍	(2017) 云0112执1594号	36	女	5335231981*...
70021...	李卫春	(2017) 沪0112执2947号	37	女	3206811980*...
70021...	刘春香	(2017) 沪0112执2947号	36	女	3206261981*...
70028...	张文影	(2017) 沪0112执5862号	33	女	3412251983*...
70028...	徐文隆	(2017) 沪0112执5781号	72	男	3604291945*...
70028...	翟玉英	(2017) 沪0112执5781号	69	女	3604291947*...

data1

案号	被执行人	被执行人地址
(2017) 沪0116执594号	任定勇	上海市金山区金山卫镇西静路三康小区
(2017) 沪0117执594号	杨峰华	上海市松江区石湖荡镇东港村1246
(2017) 沪0118执984号	上海意邦置...	上海市青浦区重固镇北青公路７５２３号
(2017) 沪0118执584号	吴刚	安徽省淮南市八公山区新庄孜团结村典
(2017) 沪0112执759号	曹刚	上海市闵行区梅陇镇曹行村街北5号18
(2017) 沪0112执759号	虞恒强	上海市宝源路209弄12号102室1316
(2017) 沪0112执758号	曹刚	上海市闵行区梅陇镇曹行村街北5号15

data2

Figure 10: *The second example of tolerant search.*

4 Query

Query search is to find related documents in *instruments* using a query string. The main point is to match the query string and all documents for a similar score.

4.1 Methodology

Our methodology is divided into three parts: (1) Construct and save term frequency for each document in advance. (2) For a coming query string, match it with all related documents to get a score. (3) Rank these related documents. Here *instruments* is needed and we use *ins_dictionary* and *ins_postings*. The implementation details are as follows:

4.1.1 Term frequency dictionary

We will construct the term frequency dictionary ins_tfdict in advance for each document in *instrument*. For all attributes of a document in *instrument*, we ignore ”id” and process the remaining ones as follows:

1. Remove all useless punctuation and spaces.
2. Use package jieba to segment all remaining attributes and remove stop words.
3. Count the term frequency.

After processing a document, we will get a term frequency dictionary like $\{term : frequency\}$. Combine these dictionaries for all documents in *instrument*, we will get ins_tfdict with docID as keys and $\{term : frequency\}$ as values.

4.1.2 Match query and documents

We use tfidf to match a query string and documents, The detailed process is as follows:

1. Process the query string: we remove all useless punctuation and spaces in the query. Then we use jieba to segment the query and remove stop words. After that, a query will be converted to a list of search keys.

2. Gather related documents: avoid matching all documents, we gather related documents in advance. For each key in the query, we traverse *ins_dictionary* to find a same term (if any) and gather its corresponding documents in *ins_postings*. After that, we will gather documents which include at least one search key in the query.

3. Calculate the final score for a related document. For a related document d and a term t, we can find term frequency information $\mathrm{tf}_{t,d}$ in ins_tfdict and document frequency $\mathrm{df_t}$ in *ins_dictionary*, so that we can easily calculate the tf.idf value of term t in document d. The final score for a related document d is the sum of tf.idf for all interacted terms in query q and d :

$$\mathrm{tf.idf}_{t,d} = (1 + \log_{10}\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)$$
$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} \mathrm{tf.idf}_{t,d}$$

4.1.3 Rank documents

In Section 4.1.2, we have generated all related documents and their final scores for a query search string, so we just need to rank these documents in descending order according to their scores now.

4.2 Result

In this section, we will show a example of query. Because there are too many results, only the first few results and related fields are displayed.

- Search **“本院依法缺席审理。本案现已审理终结。”** , and the result is shown in Figure 11.

文书类别	案由	承办部门	级别	结案日期	Content
判决书	提...	民事审...	一审	2019年05月27日	上海市徐汇区人民法院 民事判决书 案号：（2017）沪0104民初19331号 原告：王雷，男，1985年6月5日出生，汉族，户籍地...
判决书	民...	商事审...	一审	2018年07月30日	上海市静安区人民法院 民事判决书 案号：（2017）沪0106民初34180号 原告：陈家泉，男，1964年6月7日出生，汉族，住上...
判决书	医...	民事审...	一审	2018年12月10日	上海市黄浦区人民法院 民事判决书 案号：（2017）沪0101民初25826号 原告：上海市黄浦区东南医院，住所地上海市黄浦区。...
判决书	民...	民一庭	一审	2017年06月29日	上海市普陀区人民法院 民事判决书 案号：（2016）沪0107民初4820号 原告：张道舜，男，1969年4月1日出生，汉族，住江苏...
判决书	民...	民事审...	一审	2019年05月20日	上海市徐汇区人民法院 民事判决书 案号：（2019）沪0104民初1893号 原告：杨卓军，男，1972年3月12日出生，汉族，住上...
判决书	借...	民事审...	一审	2019年07月05日	上海市徐汇区人民法院 民事判决书 案号：（2019）沪0104民初10978号 原告：朱悦，男，1963年11月26日出生，汉族，住上...
判决书	生...	民事审...	一审	2019年04月15日	上海市徐汇区人民法院 民事判决书 案号：（2019）沪0104民初1334号 原告：茆长彩，女，1963年6月6日出生，汉族，户籍地...
判决书	民...	民事审...	一审	2019年07月15日	上海市徐汇区人民法院 民事判决书 案号：（2019）沪0104民初2667号 原告：陈珺，女，1977年11月26日出生，汉族，住上海...
判决书	侵...	知识产...	一审	2019年06月14日	上海市徐汇区人民法院 民事判决书 案号：（2019）沪0104民初9962号 原告：九牧王股份有限公司，住所地福建省泉州市。 ...
判决书	民...	民事审...	一审	2019年05月05日	上海市静安区人民法院 民事判决书 案号：（2019）沪0106民初10872号 原告：于旺，男，1973年5月8日出生，汉族，户籍地...

Figure 11: *The example* “本院依法缺席审理。本案现已审理终结。”.

5 Ranking

5.1 Methodology

In our system, the fields marked with green in the cell are sortable fields, and users can sort the results by clicking on the field. After obtaining the users' click, the system can sort the documents or data according to the current state of the field, including ascending and descending order.

The logic for sorting the data based on a field is shown in Figure 12, when users do not click the field, the data is in an unsorted state. After clicking, it will change to a descending order, and click again to change to an ascending state. After that, the data will switch between descending and ascending order. When users click on another field to sort the data, the status of the other sortable fields becomes unsorted.

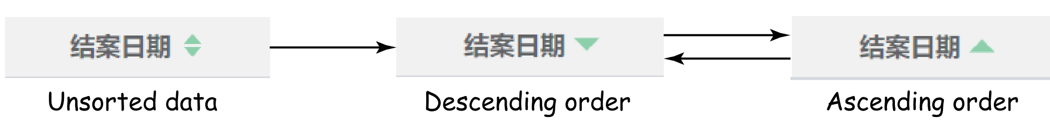

Figure 12: *The logic for sorting the data.*

For different data, we have implemented different fields that can be sorted.

- For *data1*, users can rank the results according to fields *'caseCode'*, *'regDate'* and *'publishDat'*. The header of *data1* is shown in Figure 13, and the fields enclosed by the red box are the sortable fields.

Figure 13: *The header of data1.*

- For *data2*, users can rank the results according to fields *'caseCode'* and *'imoney'*. The header of *data2* is shown in Figure 14, and the fields enclosed by the red box are the sortable fields.

Figure 14: *The header of data2.*

- For *instruments*, users can rank the results according to fields *'caseCode'* and *'closingDate'*. The header of *instruments* is shown in Figure 15, and the fields enclosed by the red box are the sortable fields.

Figure 15: *The header of instruments.*

5.2 Result

In this section, we will show an example of ranking. Since there are too many data fields, we only select three fields in *data1* for demonstration. The result is shown in Figure 16, we can find that our system has responded to the user's click and returned the correctly sorted data.

Figure 16: *An example of ranking.*

6 Conclusions

In general, our system meets the requirements of the project. The operations we have implemented include boolean search, tolerant search, query, and ranking the returned search results.

Appendices

A An Appendix of Field Description

1. Field description of *data1* is shown in Table 4:

Field	Description
id	id of this piece of data.
iname	The name of the judgment debtor.
caseCode	Case number.
age	The age of the judgment debtor.
sexy	The gender of the judgment debtor.
cardNum	Identification number/organization code.
bussinessEntity	Legal person (If the judgment debtor is a company).
courtName	Executive court.
areaName	Province.
gistId	Execution basis.
regDate	Filing time.
gistUnit	The unit that made the basis for execution.
duty	Obligations determined by the legal instruments in force.
performance	The performance of the judgment debtor.
performedPart	Fulfilled part.
unperformPart	Unfulfilled part.
disruptTypeName	Specific circumstances of the judgment debtor's behavior.
publishDate	Release time.
qysler	'cardNum', 'corporationtypename': people-company relationship, 'iname'

Table 4: *Field description of data1.*

2. Field description of *instruments* is shown in Table 5:

Field	Description
id	id of this piece of data.
caseCode	Case number.
title	The title of the instrument.
type	The type of the instrument.
cause	Cause.
department	Undertaking department.
level	Level.
closingDate	Closed date.
content	The content of the instrument.

Table 5: *Field description of instruments.*

3. Field description of *data2* is shown in Table 6:

Field	Description
caseCode	Case number.
iname	The name of the judgment debtor.
iaddress	The address of the judgment debtor.
imoney	The amount executed (RMB).
ename	The name of the execution applicant.
courtName & phone	Court, contact phone.

Table 6: *Field description of data2.*