CSCI 572 Homework2

Team Member:

Chenyin Sang, 2812623778

Dongni Zhao, 3282822727

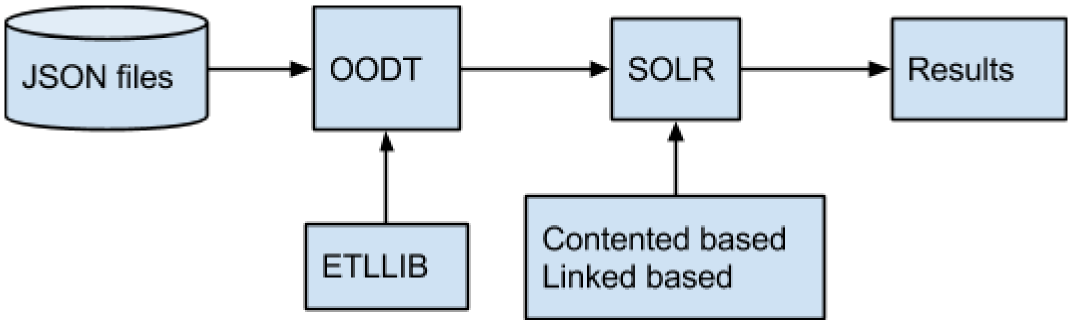
Hongzhi Li, 1322106452

Wenhai Zhu, 9262786977

1. Process

In this homework, we built an employment job search engine. The data is from the deduplicated json files from homework 1. We first built an indexing system. We combine Apache OODT and ETLLIB together and use poster function in ETLLIB to post all JSON files into SOLR. Before we post JSON files, we compute the link-based score using the method similar to PageRank and write this score as a field into the JSON files. In SOLR, we write some queries to query some keywords and see the results. Finally, we also write a python script to demonstrate the answers to the four questions.

The whole process of this search engine is like followings.



We also add 3 new meta data into the JSON files. The first one is the value of rank score of this file. The second one is the category this job belongs to. For example, this job may belong to a commercial area, business area, or industrial area. The third one is the time that this job filled in. We use “lastseendate” minus “firstseendate”. If the difference is small, it means this job is very quickly filled in.

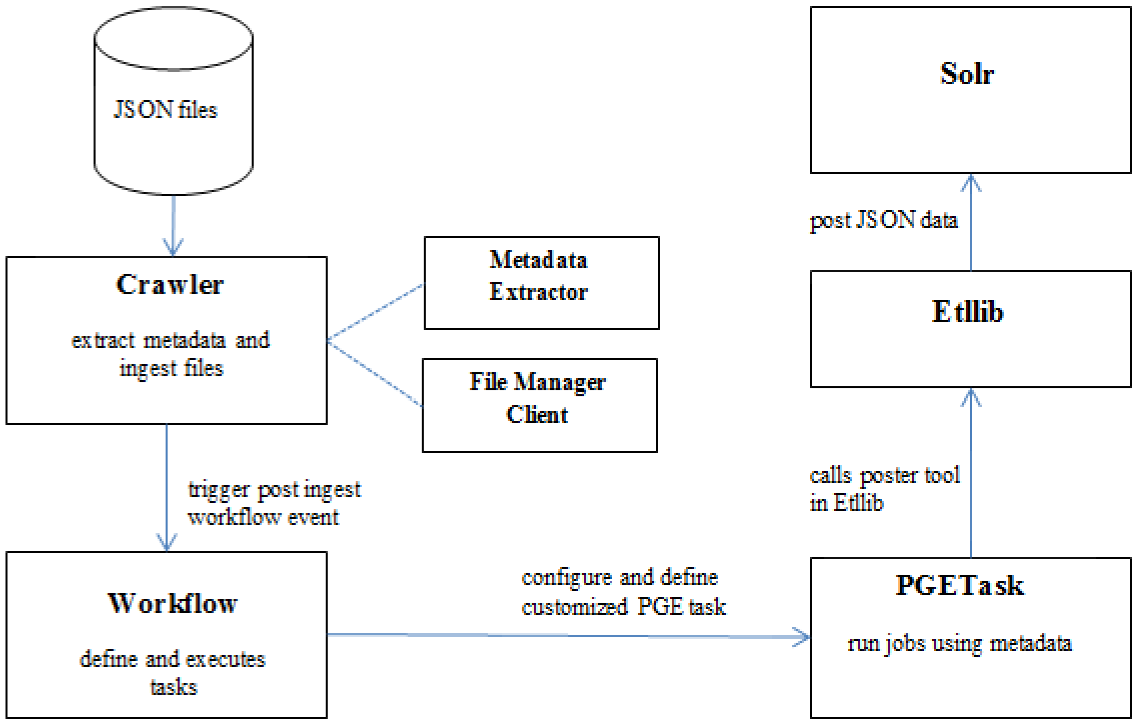
We did this homework both on a small dataset which contains **100,000 files** and a big dataset which contains **2,107,060 files**.

2. Indexing system using OODT, ETLLIB, Solr

a. Indexing process

The whole indexing process starts from OODT Crawler. The Crawler ingests all the JSON files with File Manager Client and extracts metadata from these JSON files with Metadata Extractor. Once the JSON file is ingested successfully, the Crawler triggers the post ingest workflow event with the name [ProductType]Ingest. Then the [ProductType]Ingest event calls the customized workflow which defines a series of tasks to be tackled one after one.

It is in these tasks where we execute the post action in Etllib and index all the JSON files to Solr. Below is simplified process graph:


JSON files
Crawler
extract metadata and ingest files
Metadata Extractor
File Manager Client
trigger post ingest workflow event
Workflow
define and executes tasks
configure and define customized PGE task
PGETask
run jobs using metadata
calls poster tool in Etllib
Etllib
post JSON data
Solr

Indexing Process

b. Interaction between Workflow and Etllib

As the process graph above shows, the workflow will run the PGE task which calls the poster tool in Etllib. The Etllib post function requires the Filename metadata extracted by Metadata Extractor and then prepare and post the JSON content to Solr by URL. Every successful ingestion will trigger such a workflow until all the JSON files under the working directory being traversed.

c. Pros and Cons of Etllib

Using Etllib, we can easily post JSON files to Solr. It can make sure that each file is successfully post and loop to next file. Another advantage is that it is very fast when posting JSON files into Solr.

However, there are still some aspects that ETLLIB can apply and improve itself. Firstly, when we use "poster" function, it is difficult to know how to provide files as a parameter into this function. It would be better if there is an example to illustrate. It can be also that provide a parameter into this function like "-f xx.json". Secondly, the provided command usage messages are too vague. It would be better if it provides some examples on how to use those commands. Thirdly, more detail error messages would be helpful while exception occurs.

d. Metrics Of Ingestion

During the whole process, each JSON file takes **0.0923**s to ingest into File Manager and trigger workflow event and execute tasks, and then it costs **0.0011**s to post and indexed by Solr on average, so the each job takes **0.09344**s on average. Thus obviously, the bottleneck of the whole process lies on OODT because it is time-consuming to ingest file and run triggered workflow which invokes a series of components to get its tasks done. As for the overall wall clock time, we spent about **155 minutes** for full indexing of 100k jobs and **55 hours** for full indexing of 2.1M jobs.

e. Strengths and Weakness of OODT

Strengths:

OODT is consisted of several components and each of them is highly flexible and can be easily customized. Thus, we can exploit any combination of these components based on our requirements and ignore other components. Take this assignment, we need to ingest the JSON files and index these JSON contents into Solr using the extracted metadata. So we can just customize the Crawler, File Manager and Workflow. Then these customized components will collaborate with each other perfectly according our configuration.

Weaknesses:

While tackling this assignment, we found that there is no way for OODT to adjust its behavior based on the execution result of external programs it triggers. For example, in the PGE task, we invoke the Etllib post function. However, OODT will not know whether the post action succeed or not and then adjust its behavior accord to the result. So, once the post action fails, OODT will not post this JSON file again and some JSON files may lose in Solr. OODT is easy to install but hard to understand how to configure. It took us a long time to understand how OODT works and how all these components collaborate with each other. Moreover, there are limited tutorial documents we can access currently.

3. Ranking algorithms

3.1 Content-Based:

Algorithm:

Our input is the term user's search, and we will set boost in pf under dismax.

We use the default score in solr and modify the boost according our understanding of the query data.

$$Score(q,d) = \sum_{t\ in\ q} \left(tf(t\ in\ d) \cdot idf^{2}(t) \cdot t.getBoost() \cdot norm(t,d)\right) \cdot coord(q,d) \cdot queryNorm(q)$$

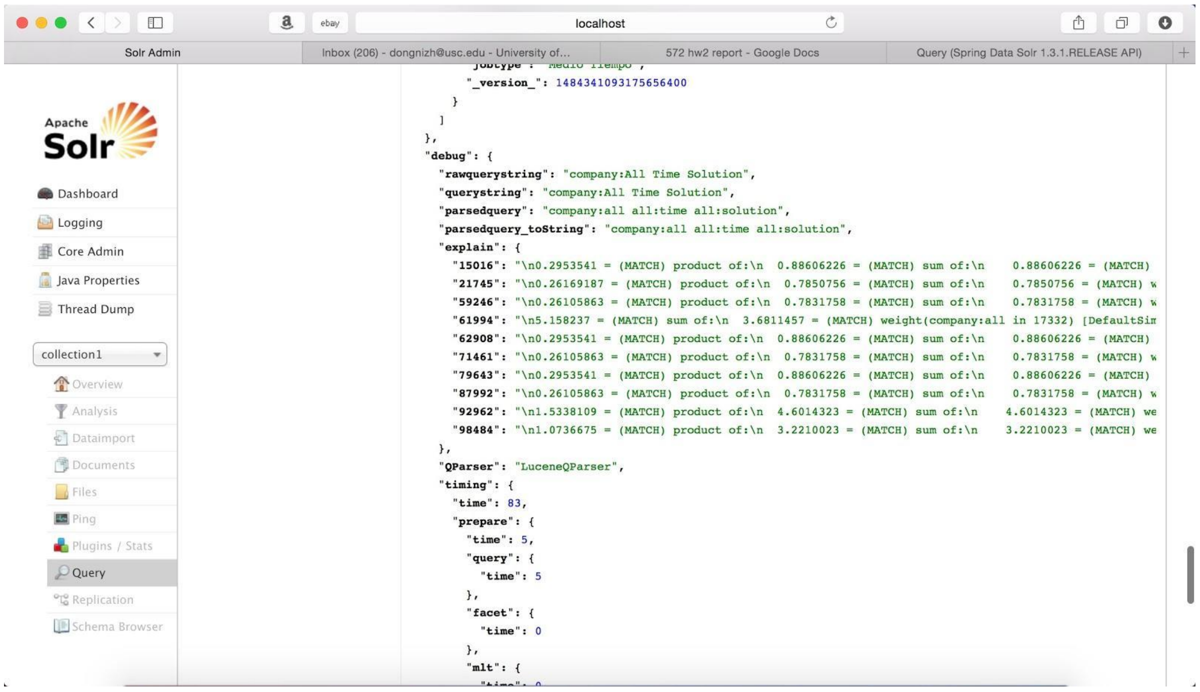
tf means term frequency, it measures of how often a particular term appears in a mathcing document and shows the relevance between the term and the document. TF-IDF is an efficent way to avoid common words happened so frequently that can mess up the real result of the searching word. Always, this way is used mostly in a huge volume of dataset to search by key words.

idf means inverse frequency, it shows how many total documents the search term appears within.

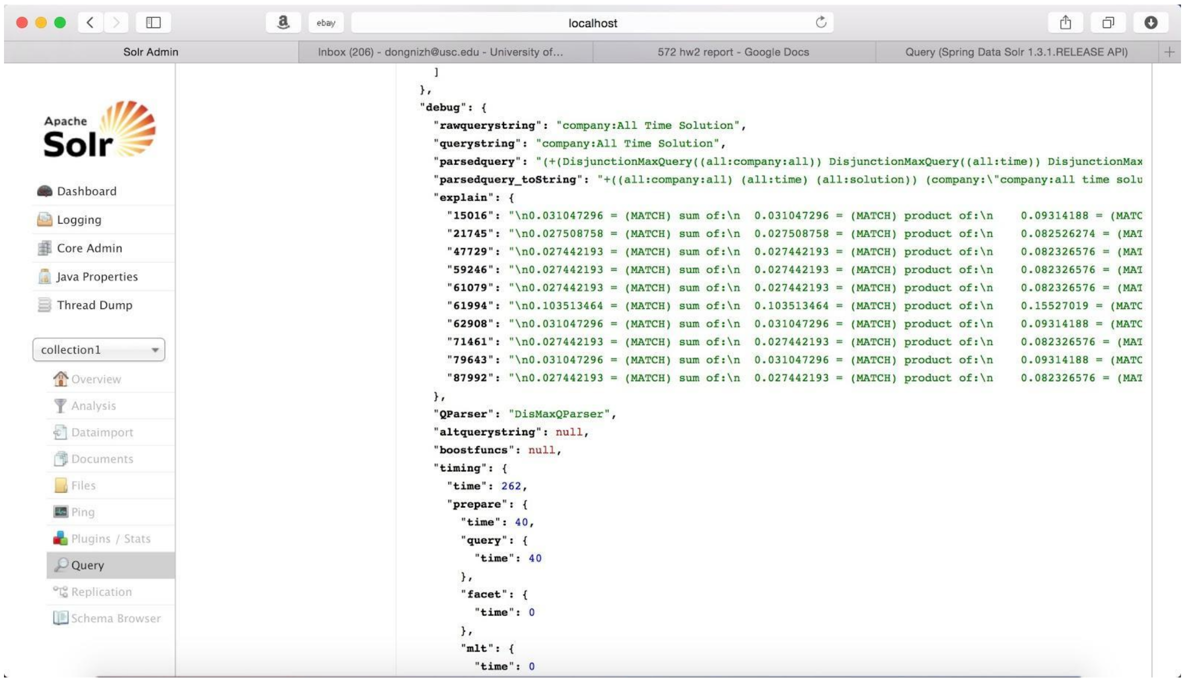
t.getBoost means term boost, it set the weight of the content based on our cognition of the importance of the fields.

$$norm(t,d) = d.getBoost() \cdot lengthNorm(f) \cdot f.getBoost()$$

norm means normalization, it calculated from the product of the document boost, the field boost and the length norm, which is encoded into a single byte stored in Solr index. Such as the result showing below:



```
        "jobtype": "Medio Tiempo",
        "_version_": 1484341093175656400
      }
    ]
  },
  "debug": {
    "rawquerystring": "company:All Time Solution",
    "querystring": "company:All Time Solution",
    "parsedquery": "company:all all:time all:solution",
    "parsedquery_toString": "company:all all:time all:solution",
    "explain": {
      "15016": "\n0.2953541 = (MATCH) product of:\n  0.88606226 = (MATCH) sum of:\n    0.88606226 = (MATCH)
      "21745": "\n0.26169187 = (MATCH) product of:\n  0.7850756 = (MATCH) sum of:\n    0.7850756 = (MATCH) w
      "59246": "\n0.26105863 = (MATCH) product of:\n  0.7831758 = (MATCH) sum of:\n    0.7831758 = (MATCH) w
      "61994": "\n5.158237 = (MATCH) sum of:\n  3.6811457 = (MATCH) weight(company:all in 17332) [DefaultSim
      "62908": "\n0.2953541 = (MATCH) product of:\n  0.88606226 = (MATCH) sum of:\n    0.88606226 = (MATCH)
      "71461": "\n0.26105863 = (MATCH) product of:\n  0.7831758 = (MATCH) sum of:\n    0.7831758 = (MATCH) w
      "79643": "\n0.2953541 = (MATCH) product of:\n  0.88606226 = (MATCH) sum of:\n    0.88606226 = (MATCH)
      "87992": "\n0.26105863 = (MATCH) product of:\n  0.7831758 = (MATCH) sum of:\n    0.7831758 = (MATCH) w
      "92962": "\n1.5338109 = (MATCH) product of:\n  4.6014323 = (MATCH) sum of:\n    4.6014323 = (MATCH) we
      "98484": "\n1.0736675 = (MATCH) product of:\n  3.2210023 = (MATCH) sum of:\n    3.2210023 = (MATCH) we
    },
    "QParser": "LuceneQParser",
    "timing": {
      "time": 83,
      "prepare": {
        "time": 5,
        "query": {
          "time": 5
        },
        "facet": {
          "time": 0
        },
        "mlt": {
          "time": 0
```



```
    ]
  },
  "debug": {
    "rawquerystring": "company:All Time Solution",
    "querystring": "company:All Time Solution",
    "parsedquery": "(+(DisjunctionMaxQuery((all:company:all)) DisjunctionMaxQuery((all:time)) DisjunctionMax
    "parsedquery_toString": "+((all:company:all) (all:time) (all:solution)) (company:\"company:all time solu
    "explain": {
      "15016": "\n0.031047296 = (MATCH) sum of:\n  0.031047296 = (MATCH) product of:\n    0.09314188 = (MATC
      "21745": "\n0.027508758 = (MATCH) sum of:\n  0.027508758 = (MATCH) product of:\n    0.082526274 = (MAT
      "47729": "\n0.027442193 = (MATCH) sum of:\n  0.027442193 = (MATCH) product of:\n    0.082326576 = (MAT
      "59246": "\n0.027442193 = (MATCH) sum of:\n  0.027442193 = (MATCH) product of:\n    0.082326576 = (MAT
      "61079": "\n0.027442193 = (MATCH) sum of:\n  0.027442193 = (MATCH) product of:\n    0.082326576 = (MAT
      "61994": "\n0.103513464 = (MATCH) sum of:\n  0.103513464 = (MATCH) product of:\n    0.15527019 = (MATC
      "62908": "\n0.031047296 = (MATCH) sum of:\n  0.031047296 = (MATCH) product of:\n    0.09314188 = (MATC
      "71461": "\n0.027442193 = (MATCH) sum of:\n  0.027442193 = (MATCH) product of:\n    0.082326576 = (MAT
      "79643": "\n0.031047296 = (MATCH) sum of:\n  0.031047296 = (MATCH) product of:\n    0.09314188 = (MATC
      "87992": "\n0.027442193 = (MATCH) sum of:\n  0.027442193 = (MATCH) product of:\n    0.082326576 = (MAT
    },
    "QParser": "DisMaxQParser",
    "altquerystring": null,
    "boostfuncs": null,
    "timing": {
      "time": 262,
      "prepare": {
        "time": 40,
        "query": {
          "time": 40
        },
        "facet": {
          "time": 0
        },
        "mlt": {
          "time": 0
```

queryNorm means query normalization, it attempt to make scores between queries comparable and it does not affect the overall relevancy ordering.

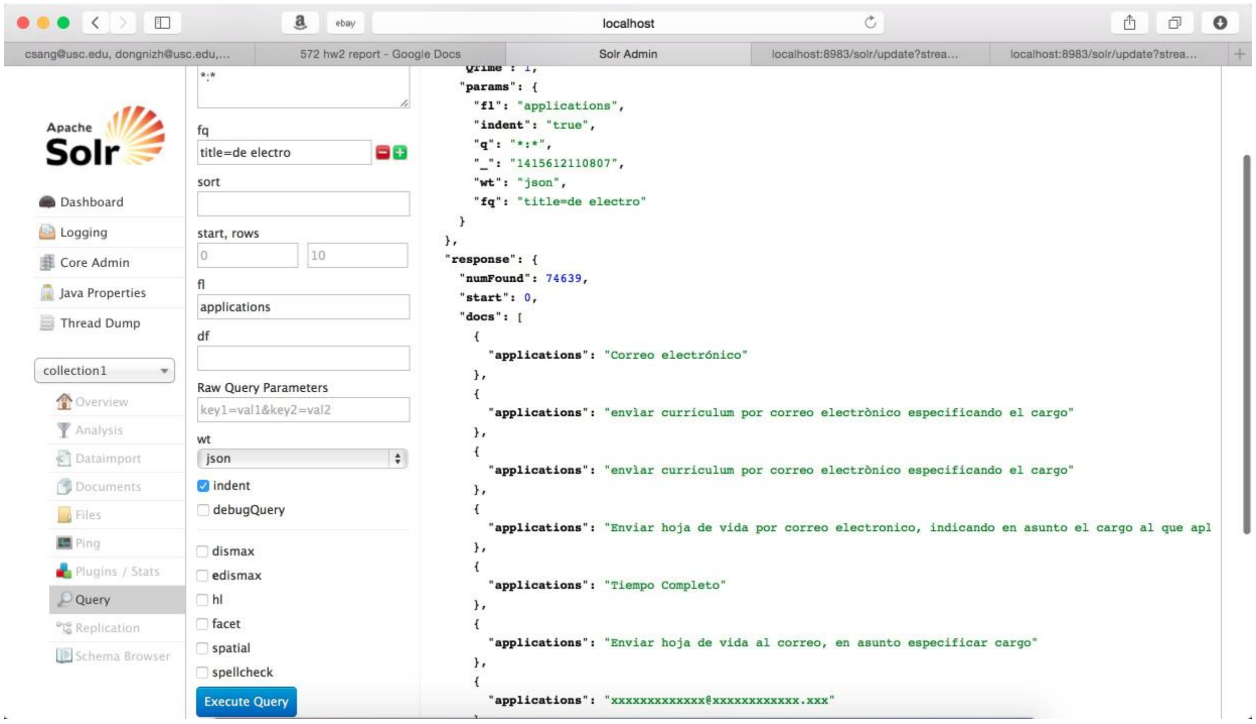
coord means coordination factor, it shows how much of the query each document matches.

Moreover, since we have already analysed the content of each column, we define the most suitable field type for the fields. Also when we do query, we use edit distance which Solr has already provided, so that we can use query: string~ to execute the string similarity.

Input: input json files

Test and Prove:

When we test two keywords: "de" and "electro", although "de" appears many more times than "electro", since we use the square of idf(t), the score of "de" is relatively low. This equation balance the recall and precision. In the result as we showing below, the rank of "electro" is higher than "de", which accords our guess.

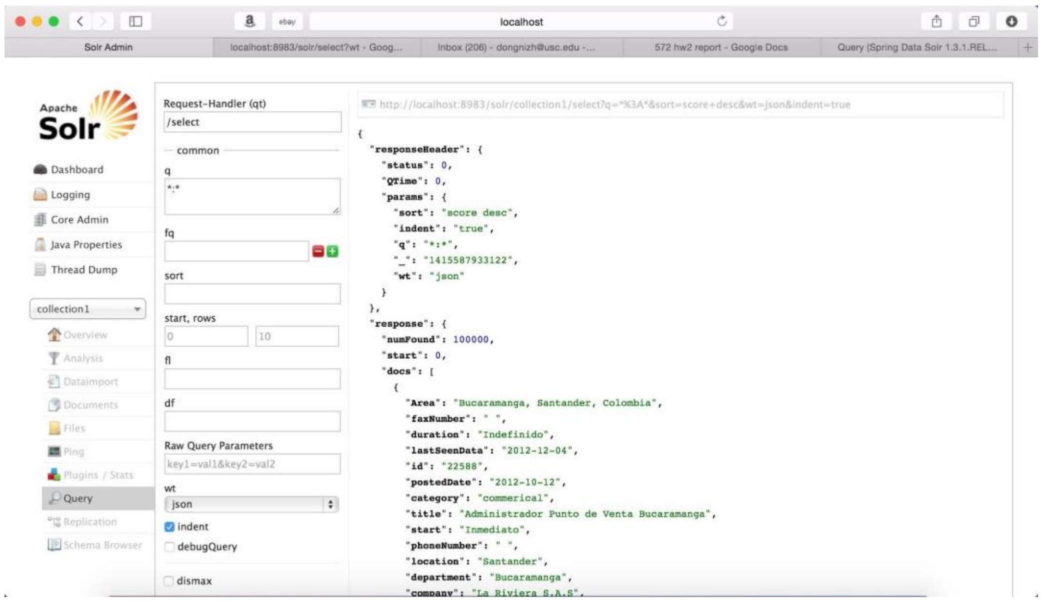


q: *:*
fq: title=de electro
sort:
start, rows: 0 10
fl: applications
df:
Raw Query Parameters: key1=val1&key2=val2
wt: json
[✓] indent
[ ] debugQuery

```
      "params": {
        "fl": "applications",
        "indent": "true",
        "q": "*:*",
        "_": "1415612110807",
        "wt": "json",
        "fq": "title=de electro"
      }
    },
    "response": {
      "numFound": 74639,
      "start": 0,
      "docs": [
        {
          "applications": "Correo electrónico"
        },
        {
          "applications": "envìar curriculum por correo electrònico especificando el cargo"
        },
        {
          "applications": "envìar curriculum por correo electrònico especificando el cargo"
        },
        {
          "applications": "Enviar hoja de vida por correo electronico, indicando en asunto el cargo al que apl
        },
        {
          "applications": "Tiempo Completo"
        },
        {
          "applications": "Enviar hoja de vida al correo, en asunto especificar cargo"
        },
        {
          "applications": "xxxxxxxxxxxxx@xxxxxxxxxxxx.xxx"
```

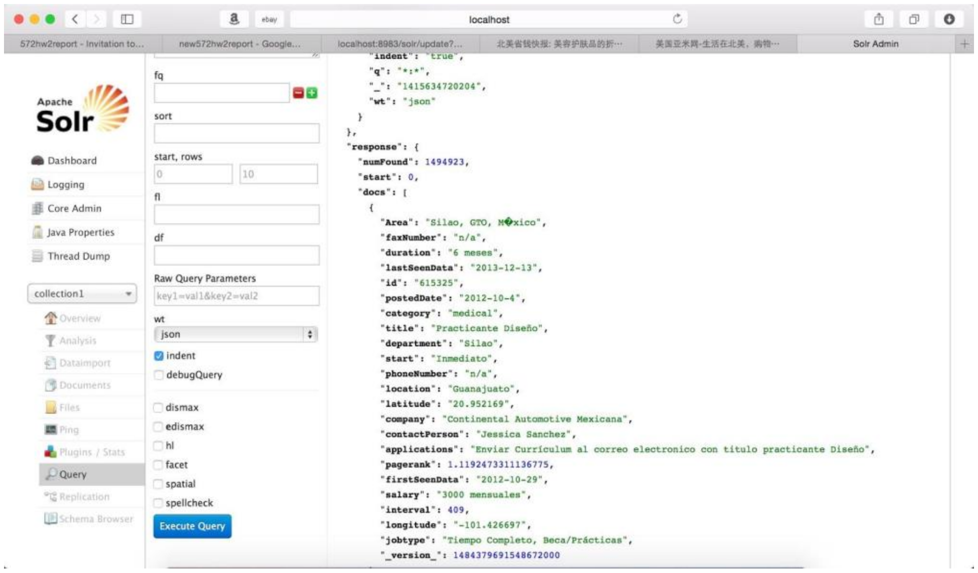
Result:

For small dataset (100,000): As this dataset is small, we posted this dataset about 5-6 times, and finally all 100,000 succeed.



```
http://localhost:8983/solr/collection1/select?q=*%3A*&sort=score+desc&wt=json&indent=true
{
  "responseHeader": {
    "status": 0,
    "QTime": 0,
    "params": {
      "sort": "score desc",
      "indent": "true",
      "q": "*:*",
      "_": "1415587933122",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 100000,
    "start": 0,
    "docs": [
      {
        "Area": "Bucaramanga, Santander, Colombia",
        "faxNumber": " ",
        "duration": "Indefinido",
        "lastSeenData": "2012-12-04",
        "id": "22588",
        "postedDate": "2012-10-12",
        "category": "commerical",
        "title": "Administrador Punto de Venta Bucaramanga",
        "start": "Inmediato",
        "phoneNumber": " ",
        "location": "Santander",
        "department": "Bucaramanga",
        "company": "La Riviera S.A.S",
```

For big dataset(2,107,060): Because of the bottleneck of OODT, we tried to post 2,107,060, but only 1,494,923 succeed. And this took a long time to post, so we only posted it once.



```
      "indent": "true",
      "q": "*:*",
      "_": "1415634720204",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 1494923,
    "start": 0,
    "docs": [
      {
        "Area": "Silao, GTO, M�xico",
        "faxNumber": "n/a",
        "duration": "6 meses",
        "lastSeenData": "2013-12-13",
        "id": "615325",
        "postedDate": "2012-10-4",
        "category": "medical",
        "title": "Practicante Diseño",
        "department": "Silao",
        "start": "Inmediato",
        "phoneNumber": "n/a",
        "location": "Guanajuato",
        "latitude": "20.952169",
        "company": "Continental Automotive Mexicana",
        "contactPerson": "Jessica Sanchez",
        "applications": "Enviar Currículum al correo electronico con titulo practicante Diseño",
        "pagerank": 1.1192473311136775,
        "firstSeenData": "2012-10-29",
        "salary": "3000 mensuales",
        "interval": 409,
        "longitude": "-101.426697",
        "jobtype": "Tiempo Completo, Beca/Prácticas",
        "_version_": 1484379691548672000
```

3.2 Link-based

Algorithm:

We extract useful attribute as the keywords, and then put it into a hashmap, then we use this hashmap to make sure the connection index of each file and build a graph with links of two files share the same attribute. Using the equation given by GOOGLE, we iterated each pagerank 20 times in order to get a stable result.

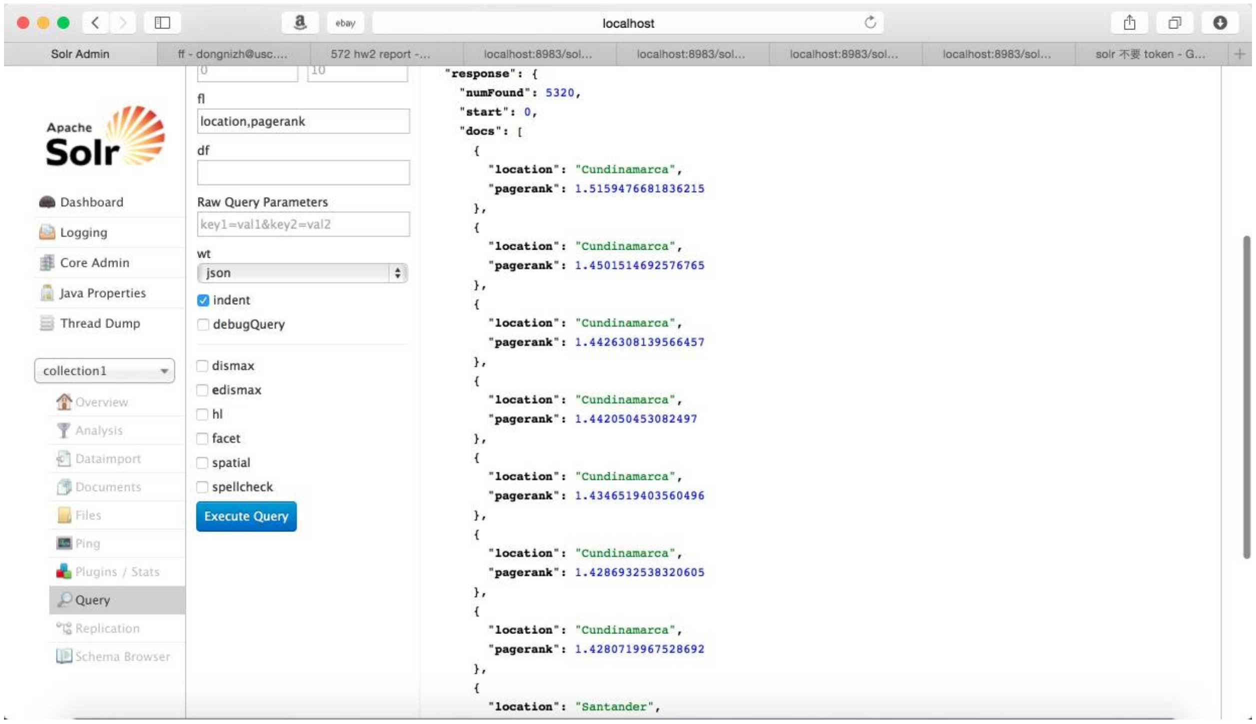
Build inverted index hashmap according to the attributes value(Using three aspects: 1. company name, 2. title AND location, 3. jobType AND location) which file contains it. Firstly, company name is not the unique value in the dataset and it shows relevance with other data. Secondly, due to the first question, we need the result based on title and location, so that we choose this combination as an inverted index. The same as the question 4, we need job type within the location.

In order to save memory, we build the graph every 100 files and then clean hashmap. From the hashmap to save memory, then we use this <key, value >as <attribute value, pagerank> to build the graph.

From the graph, compute the pagerank iteratively by using original pagerank function: $p(A) = (1-d) + d(p(B)/c(B) + p(c)/c(c)+\dots)$ right now, we are using 20 times iteration to get the result. This function is in the PageRank.java, named as computePageRank function.

Input: input json files

Test and Prove: We just query one title name. Based on our algorithm, when title and location are the same, the rank will be really high. So we test as the figure we showing below, the results match our guess.


fl: location,pagerank

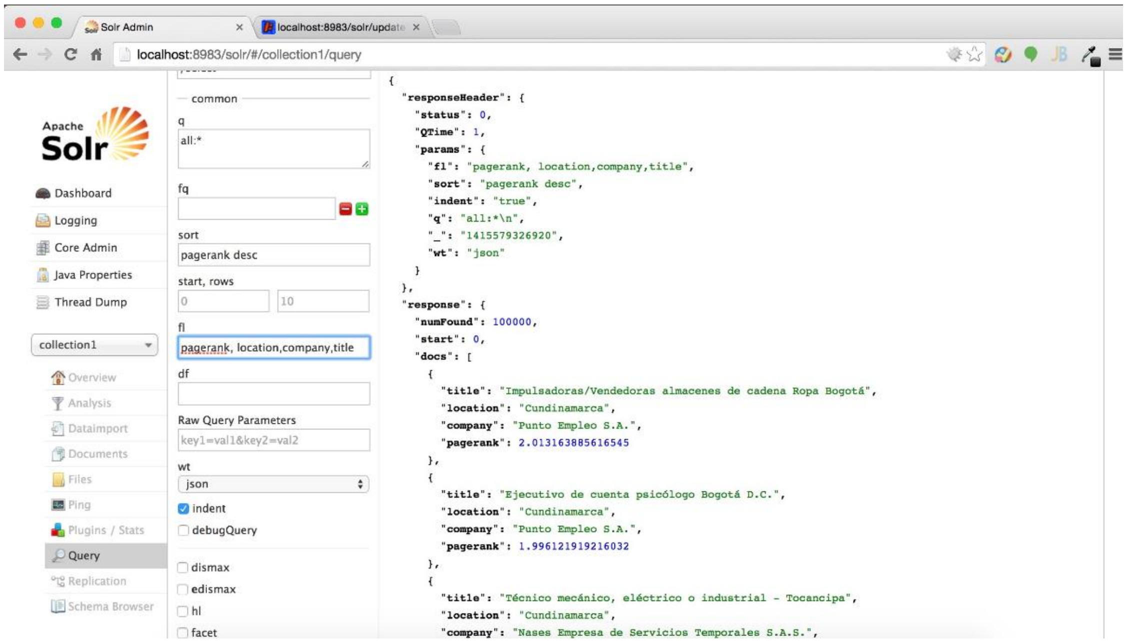
wt: json

[✓ indent]

```
"response": {
  "numFound": 5320,
  "start": 0,
  "docs": [
    {
      "location": "Cundinamarca",
      "pagerank": 1.5159476681836215
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.4501514692576765
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.4426308139566457
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.442050453082497
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.4346519403560496
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.4286932538320605
    },
    {
      "location": "Cundinamarca",
      "pagerank": 1.4280719967528692
    },
    {
      "location": "Santander",
```

Result:

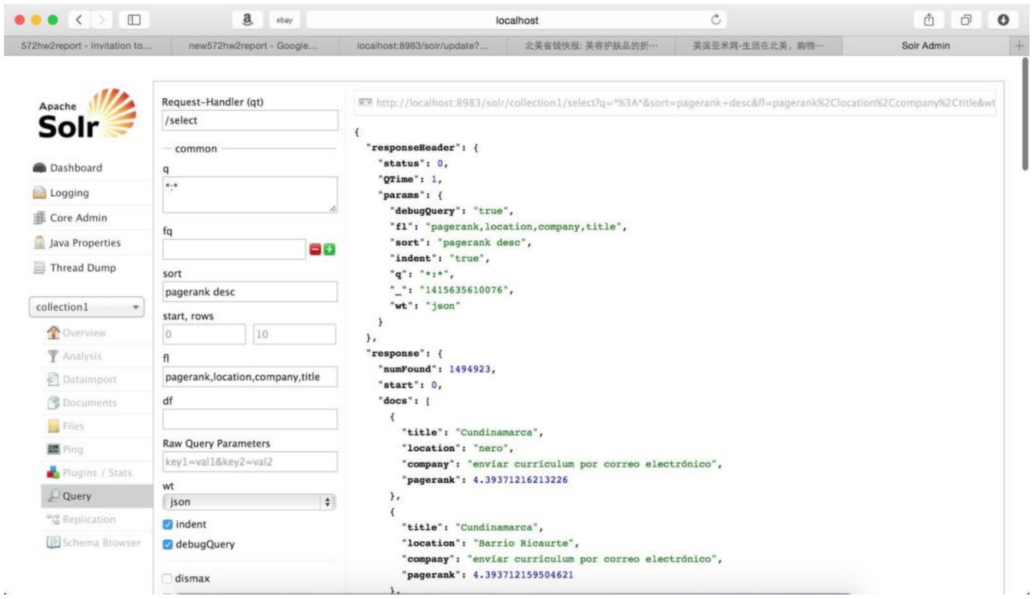
For small dataset (100,000): As this dataset is small, we posted this dataset about 5-6 times, and finally all 100,000 succeed.


localhost:8983/solr/#/collection1/query

q: all:*
sort: pagerank desc
start, rows: 0 10
fl: pagerank, location,company,title
wt: json
indent ✓

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "fl": "pagerank, location,company,title",
      "sort": "pagerank desc",
      "indent": "true",
      "q": "all:*\n",
      "_": "1415579326920",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 100000,
    "start": 0,
    "docs": [
      {
        "title": "Impulsadoras/Vendedoras almacenes de cadena Ropa Bogotá",
        "location": "Cundinamarca",
        "company": "Punto Empleo S.A.",
        "pagerank": 2.013163885616545
      },
      {
        "title": "Ejecutivo de cuenta psicólogo Bogotá D.C.",
        "location": "Cundinamarca",
        "company": "Punto Empleo S.A.",
        "pagerank": 1.996121919216032
      },
      {
        "title": "Técnico mecánico, eléctrico o industrial - Tocancipa",
        "location": "Cundinamarca",
        "company": "Nases Empresa de Servicios Temporales S.A.S.",
```

For big dataset(2,107,060): Because of the bottleneck of OODT, we tried to post 2,107,060, but only 1,494,923 succeed. And this took a long time to post, so we only posted it once.


http://localhost:8983/solr/collection1/select?q=*%3A*&sort=pagerank+desc&fl=pagerank%2Clocation%2Ccompany%2Ctitle&wt

Request-Handler (qt): /select
q: *:*
sort: pagerank desc
start, rows: 0 10
fl: pagerank,location,company,title
wt: json
indent ✓
debugQuery ✓

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "debugQuery": "true",
      "fl": "pagerank,location,company,title",
      "sort": "pagerank desc",
      "indent": "true",
      "q": "*:*",
      "_": "1415635610076",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 1494923,
    "start": 0,
    "docs": [
      {
        "title": "Cundinamarca",
        "location": "nero",
        "company": "envíar currículum por correo electrónico",
        "pagerank": 4.39371216213226
      },
      {
        "title": "Cundinamarca",
        "location": "Barrio Ricaurte",
        "company": "envíar currículum por correo electrónico",
        "pagerank": 4.393712159504621
```

3.3 Compare

a. How effective was the link-based algorithm, compared to the content based ranking algorithm?

First, link based algorithm is useful for our designed or customed queries, while content based algorithm is useful for searching all kinds of keywords. According to our query, we use location, title and company to compute our link based rank score. When we use the value in company, title or location field, link based algorithm can show the result directly. Secondly, the link based algorithm shows results as a qualification of relevance instead of by using the score of TF-IDF algorithm, which is the main algorithm for content-based. In this way, we can make a guess of relevance for users, which seems like a more crucial way to provide the most likely information.

b. What challenge questions were more appropriate for the link based algorithm compared to the content one?

Link based algorithm can be used to predict the job types in the future in one location, since we use location, title and company to compute the rank scores. So, those queries that search locations, titles and companies can use the link based algorithm, because it considered the relevance of these three fields through entire documents.

4. Queries to four questions:

a. Predict which geospatial areas will have which job types in the future.

This question is asking us to find which job types will occur in the future for a certain area. So what we do is to group by the "location", and find the most job types for now. Just select the first data sorting by pagerank, since link-based page rank can show the relevance of location and title. It is reasonable to assume that if there are more certain jobs now, these jobs will be still more in the future.

The query is:

http://localhost:8983/solr/select?wt=python&indent=true&q=*:*&group=true&group.limit=1&group.field=location&fl=title&group.sort=pagerank%20desc

b. Compare jobs in terms of quickly they're filled specifically in regards to region.

For this questions, we added a new field "interval" into the json to represent how quick this job is being filled in. We use the difference between "lastseendate" and "postedDate". If this job is filled in small days, it means it filled in very quickly. We assume the difference is at least 1 day, because when a company post the job, it is not likely that the job is filled in the same day. It needs time to apply. We just select the data whose pagerank is highest, since it can show the relevance between titles and locations.

The query is:

http://localhost:8983/solr/select?wt=python&indent=true&q=*:*&fq=interval:[1%20TO%20*]&group=true&group.limit=1&group.field=location&fl=title,interval&group.sort=interval%20asc,pagerank%20desc

c. Can you classify and zone cities based on the jobs data (E.G. commercial shopping region, industrial, residential, business offices, medical, etc)?

We added a new field "category" into the json file. So we can easily classify the job into different area using SOLR. The process to compute the category is as followings. We first use SOLR to find all job titles and sort by their frequencies. Then we classify the keywords into 7 areas manually. The 7 areas are commercial, industrial, education, medical, business, and service. Then for each job, if we find the keyword that belongs to a certain area, we will tag this job into this area.

The query is:

http://localhost:8983/solr/select/?q=*%3A*&rows=0&facet=true&facet.pivot=location,category&wt=python&indent=on

d. What are the trends as it relates to full time vs part time employment in South America?

For this question, we can compute how many full time jobs and part time jobs in the whole dataset for each date. So we can easily find the trend from the numbers.

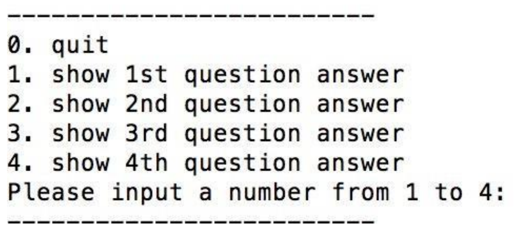
The query is:

http://localhost:8983/solr/select/?q=*%3A*&rows=0&facet=true&facet.pivot=firstSeenData,jobtype&wt=json&indent=on&fq=firstSeenData:[0000-00-00%20TO%202014-12-31]&facet.limit=100

5. Demonstrate answers

We wrote a python program to demonstrate our answers to the four questions above.

Our program is in command line. You can run it by going into the directory and type "python demonstrate.py". It will show:

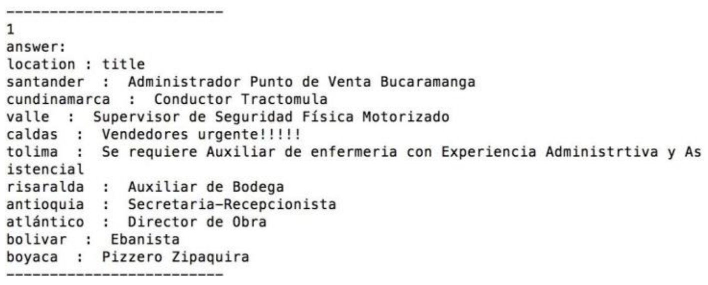


```
-------------------------
0. quit
1. show 1st question answer
2. show 2nd question answer
3. show 3rd question answer
4. show 4th question answer
Please input a number from 1 to 4:
-------------------------
```

If you input 0, it will quit the program.

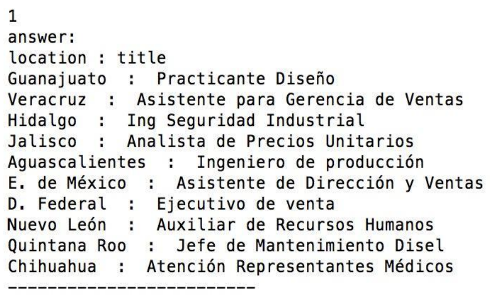
If you input 1, it will show the answers to the question 1. The following picture is the answer to the small dataset. For this question, we assume the "job types" means "job titles". We only get the top 10 results from the json result. For example, "santander" area will have more jobs about "Administrador Punto de Venta Bucaramanga" in the future. "cundinamarca" will have " Conductor Tractomula" job in the future.

For small dataset:



```
-------------------------
1
answer:
location : title
santander  :  Administrador Punto de Venta Bucaramanga
cundinamarca  :  Conductor Tractomula
valle  :  Supervisor de Seguridad Física Motorizado
caldas  :  Vendedores urgente!!!!!
tolima  :  Se requiere Auxiliar de enfermeria con Experiencia Administrtiva y As
istencial
risaralda  :  Auxiliar de Bodega
antioquia  :  Secretaria-Recepcionista
atlántico  :  Director de Obra
bolivar  :  Ebanista
boyaca  :  Pizzero Zipaquira
-------------------------
```

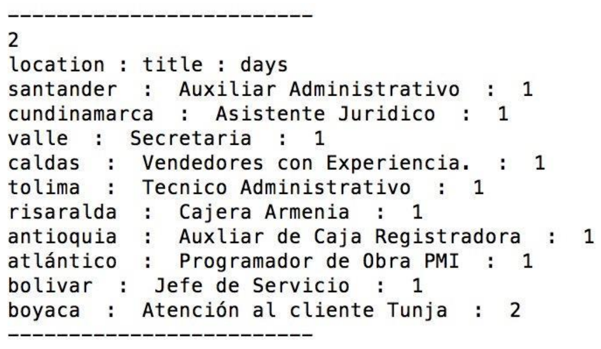
For large dataset:



```
1
answer:
location : title
Guanajuato  :  Practicante Diseño
Veracruz  :  Asistente para Gerencia de Ventas
Hidalgo  :  Ing Seguridad Industrial
Jalisco  :  Analista de Precios Unitarios
Aguascalientes  :  Ingeniero de producción
E. de México  :  Asistente de Dirección y Ventas
D. Federal  :  Ejecutivo de venta
Nuevo León  :  Auxiliar de Recursos Humanos
Quintana Roo  :  Jefe de Mantenimiento Disel
Chihuahua  :  Atención Representantes Médicos
-------------------------
```

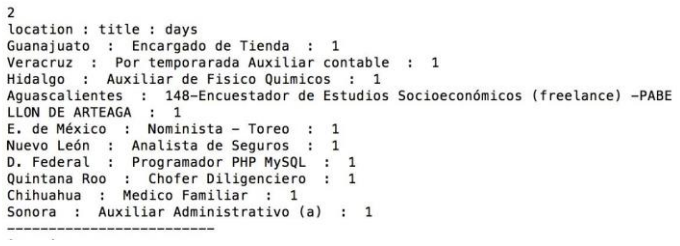
If you input 2, it will show the answers to question 2. We assume the difference is at least 1 day, because when a company post the job, it is not likely that the job is filled in the same day. It needs time to apply.

For small dataset:



```
-------------------------
2
location : title : days
santander  :  Auxiliar Administrativo  :  1
cundinamarca  :  Asistente Juridico  :  1
valle  :  Secretaria  :  1
caldas  :  Vendedores con Experiencia.  :  1
tolima  :  Tecnico Administrativo  :  1
risaralda  :  Cajera Armenia  :  1
antioquia  :  Auxliar de Caja Registradora  :  1
atlántico  :  Programador de Obra PMI  :  1
bolivar  :  Jefe de Servicio  :  1
boyaca  :  Atención al cliente Tunja  :  2
-------------------------
```

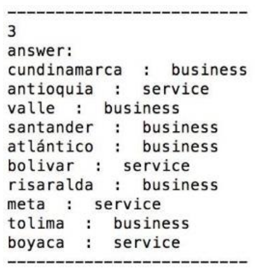
For large dataset:



```
2
location : title : days
Guanajuato  :  Encargado de Tienda  :  1
Veracruz  :  Por temporarada Auxiliar contable  :  1
Hidalgo  :  Auxiliar de Fisico Quimicos  :  1
Aguascalientes  :  148-Encuestador de Estudios Socioeconómicos (freelance) -PABE
LLON DE ARTEAGA  :  1
E. de México  :  Nominista - Toreo  :  1
Nuevo León  :  Analista de Seguros  :  1
D. Federal  :  Programador PHP MySQL  :  1
Quintana Roo  :  Chofer Diligenciero  :  1
Chihuahua  :  Medico Familiar  :  1
Sonora  :  Auxiliar Administrativo (a)  :  1
-------------------------
```

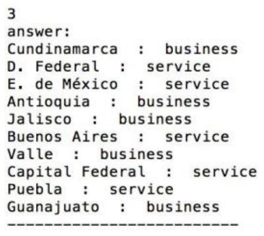
If you input 3, it will show the answers to question 3. The following is the top 10 results. For example, for "cundinamarca", it belongs to business area. For "antioquia", it belongs to "service" area.

For small dataset:



```
-------------------------
3
answer:
cundinamarca  :  business
antioquia  :  service
valle  :  business
santander  :  business
atlántico  :  business
bolivar  :  service
risaralda  :  business
meta  :  service
tolima  :  business
boyaca  :  service
-------------------------
```

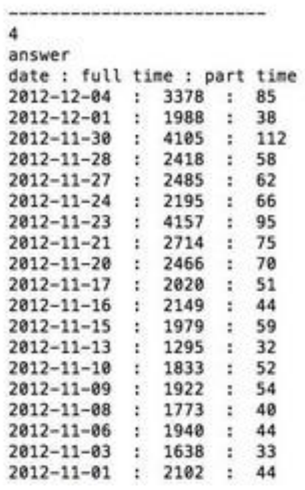
For large dataset:



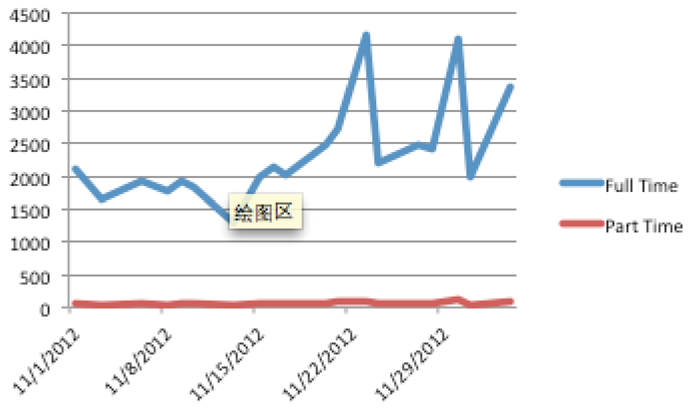
```
3
answer:
Cundinamarca  :  business
D. Federal  :  service
E. de México  :  service
Antioquia  :  business
Jalisco  :  business
Buenos Aires  :  service
Valle  :  business
Capital Federal  :  service
Puebla  :  service
Guanajuato  :  business
-------------------------
```

If you input 4, it will show the answers to question 4. We can see that the trend of full-time job in South America area is not stable. For some period, it was increased, but for some period, it decrease. The trend of the part time job i s also not stable. But in total, there are more full time jobs than part time jobs.

For small dataset:

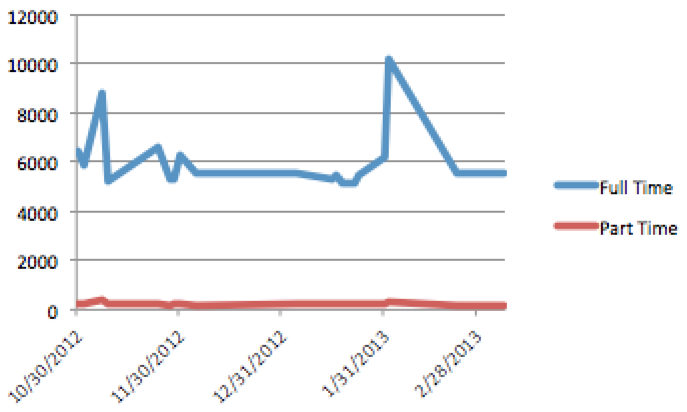


```
-------------------------
4
answer
date : full time : part time
2012-12-04  :  3378  :  85
2012-12-01  :  1988  :  38
2012-11-30  :  4105  :  112
2012-11-28  :  2418  :  58
2012-11-27  :  2485  :  62
2012-11-24  :  2195  :  66
2012-11-23  :  4157  :  95
2012-11-21  :  2714  :  75
2012-11-20  :  2466  :  70
2012-11-17  :  2020  :  51
2012-11-16  :  2149  :  44
2012-11-15  :  1979  :  59
2012-11-13  :  1295  :  32
2012-11-10  :  1833  :  52
2012-11-09  :  1922  :  54
2012-11-08  :  1773  :  40
2012-11-06  :  1940  :  44
2012-11-03  :  1638  :  33
2012-11-01  :  2102  :  44
```



For large dataset:

```
4
answer
date : full time : part time
2013-03-08  :  5535  :  165
2013-02-22  :  5551  :  195
2013-02-01  :  10206  :  311
2013-01-31  :  6190  :  251
2013-01-23  :  5450  :  207
2013-01-22  :  5135  :  217
2013-01-18  :  5170  :  217
2013-01-16  :  5467  :  222
2013-01-15  :  5323  :  215
2013-01-04  :  5574  :  204
2012-12-05  :  5546  :  190
2012-11-30  :  6283  :  212
2012-11-28  :  5298  :  215
2012-11-27  :  5301  :  195
2012-11-23  :  6634  :  222
2012-11-08  :  5255  :  215
2012-11-06  :  8791  :  365
2012-11-01  :  5844  :  220
2012-10-30  :  6483  :  234
```


10/30/2012
11/30/2012
12/31/2012
1/31/2013
2/28/2013
Full Time
Part Time