CSCI572: Assignment 2: Apache-OODT and Solr

Neha Mundada
Department of Computer Science
University of Southern California
Email: mundada@usc.edu

Shrikanth Narayanan
Department of Computer Science
University of Southern California
Email: nara471@usc.edu

Saurabh Chaure
Department of Computer Science
University of Southern California
Email: chaure@usc.edu

I. Objective

The objective of this assignment is to build both an indexing system (inverted index), and a set of ranking algorithms for the XDATA employment dataset. Once developed, we will use your newly generated search index and ranking algorithms to answer a suite of queries or Challenge Questions which will illustrate your correctly constructed search engine, and the efficacy and effectiveness of your content-based and your link-analysis based ranking algorithms. The queries that your system will answer range from per job record (and large scale) information, to per multiple job record comparisons and rankings.

II. Data Preprocessing

We used the entire 40GB data set to generate the 1.9 million json files in Assignment #1. In order to save disk space and speed up the process of accessing the json files, we divided the json files into multiple directories where each directory contained 30,000 json files. The directories were compressed using standard gzip compression format. We started the assignment#2 from this stage. We performed a number of preprocessing tasks as follows.

A. Geo-tagging the files

We used the location information, i.e., the latitude and longitude values from the job files to identify the complete address. To reverse geocode the point information, we used Yahoo Boss Geo Place Finder API [1] [2]. By using the YQL query - **select * from geo.placefinder where text="37.416275,-122.025092" and gflags="R"**, we get an xml/json response having some of the following fields.

```
<Result>
        <quality>99</quality>
        <latitude>-32.9460568</latitude>
        <longitude>-68.8184028,</longitude>
        <radius>500</radius>
        <street>Benjam n  Franklin</street>
        <postal></postal>
        <city>Las  Tortugas</city>
        <county>Godoy  Cruz</county>
        <state>Mendoza</state>
        <country>Argentina</country>
        <countrycode>AR</countrycode>
        <statecode></statecode>
</Result>
```

[1] https://developer.yahoo.com/boss/geo/docs/free_YQL.html
[2] Yahoo BOSS Geo Services : https://developer.yahoo.com/boss/geo/

B. Data Cleaning

The data in the json fields were very noisy had hence we decided to clean some parts to improve the seraching accuracy.

1) SOLR date formatting: We had to process some of the data fields in order to make it compatible with SOLR[3] indexer. All date fields had to be converted to SOLR compatible format. The given date format is YYYY-MM-DD, SOLR needs it to be in UTF format - *1995-12-31T23:59:59Z*. We referred the following apache documentation. [4] for this information.

2) Strip all fields of trailing spaces: All additional spaces before and after a field had to be striped off.

3) Handling special characters: Replace SOLR sensitive special characters, i.e., when using facet queries and performing a group by on a field, SOLR produces incorrect results if the value in the group by field has a space or a comma as per this discussion on stackoverflow[5]

C. Final JSON format

All the data transformations were done using python scripts. The final json looked like this

```
{
    "id": "000a752c-305c-4bdf-9b12-fb99d465231a",
    "Title": "jefe_de_planta_industria_alimenticia",
    "faxNumber": "",
    "Location": "Santa Fe",
    "duration": "Efectivo",
    "lastSeenDate": "2012-11-06T00:00:00Z",
    "postedDate": "2012-10-31T00:00:00Z",
    "Location2": "Santa Fe, Nuevo M xico , EEUU",
    "start": "Inmediato",
    "phoneNumber": "",
    "department": "Santa Fe",
    "hash": "9440e7ccf74e24da385c60ab031a19d032308798"
    "company": "Lic Monica Juncos GIO",
    "contactPerson": "Monica Juncos",
    "applications": "por correo electr nico",
    "firstSeenDate": "2012-11-01T00:00:00Z",
    "salary": "A convenir",
    "longitude": "-105.937799",
    "latitude": "35.6869752",
    "url": "http://www.computrabajo.com.ar/bt-ofrd-mju
    "jobtype": "tiempo_completo",
    "county": "Santa Fe County",
    "street": "Old Santa Fe Trl",
    "city": "santa_fe",
```

[3] http://lucene.apache.org/solr/
[4] http://lucene.apache.org/solr/4_10_2/solr-core/org/apache/solr/schema/DateField.html
[5] http://stackoverflow.com/questions/5790180/solr-query-with-white-space

```
    "state": "new_mexico",
    "zipcode": "87501-2009",
    "country": "united_states",
    "pagerank": 0.0000648434809387
}
```


III. Apache OODT Setup

A. Trial 1

We started by downloading the oodt source and following the individual module tutorial from the the wiki page [6]. First we installed the file-manager module by following this link [7]. We could run all the commands and query the ingested metadata. Then installed the crawler module from the link [8]. We used the MetExtractorProductCrawler crawler and configured it to use our python script which will be invoked by the *launchMetCrawler* command. To test the crawler, we ran it over a subset of the json files (1000 files) and it could successfully capture all files of type *.json.

B. Trial 2

We followed the radix installer [9] steps to setup the entire workflow process. Then using the CAS_PGE tutorial [10], we tried to setup the file-concatenator [11] example. Following were the issues faced:

1) After modifying the source code for file-concatenator to make it compatible with OODT 0.7, the oodt gives exception for MetKeys not found. This is because the filemanager jar is not present in the the workflow manager lib and res manager directory.
2) The resource manager port number in the workflow.properties for workflow manager is incorrectly set to 9300. We need to set it to 9002.
3) If the PGE_HOME variable is not set, on running the file-concatenator, it fails to pick up the PGE-Config.xml file.

[6]https://cwiki.apache.org/confluence/display/OODT/Catalog+and+Archive
[7]https://cwiki.apache.org/confluence/display/OODT/OODT+Filemgr+User+Guide
[8]https://cwiki.apache.org/confluence/display/OODT/OODT+Crawler+Help
[9]https://cwiki.apache.org/confluence/display/OODT/RADiX+Powered+By+OODT
[10]https://cwiki.apache.org/confluence/display/OODT/CAS-PGE+Learn+by+Example
[11]https://cwiki.apache.org/confluence/display/OODT/CAS-PGE+Learn+by+Example

IV. Ingesting data into SOLR using ETLLib

After successfully setting up the file-concatenator, we could run the our etllib commands to load the json files into SOLR. We used following command in the Pge-config file to be invoked by the workflow manager: *'find ¡dir¿ -name '*.json' — python poster.py -u ¡url¿'* This command generates the list of json files and passes it as an array to ETLlib potser.py command. This poster.py command puts injects data into SOLR for indexing.

A. Metrics about the ingestion into SOLR using OODT.

40 Gb data		4 Gb data	
Total Files Ingested	1956461	Total Files Ingested	263523
Total time	38 hr, 4 min	Total time	5.12 hour
Avg. time for each file	0.0702 sec	Avg. time for each file	0.0702 sec
Wall clock timing	2282.53min	Wall clock timing	307.44 min

Fig. 3: Metrics about the ingestion into SOLR using OODT

How long did each job take on average: 0.07 seconds *What were the bottlenecks?*

1) It was slow
2) It was difficult to get it work in a first go

In order to get the overall wall clock time for full indexing, we ran this command: *'/wmgr-client –url http://localhost:9001 –operation –getWorkflowInsts'*

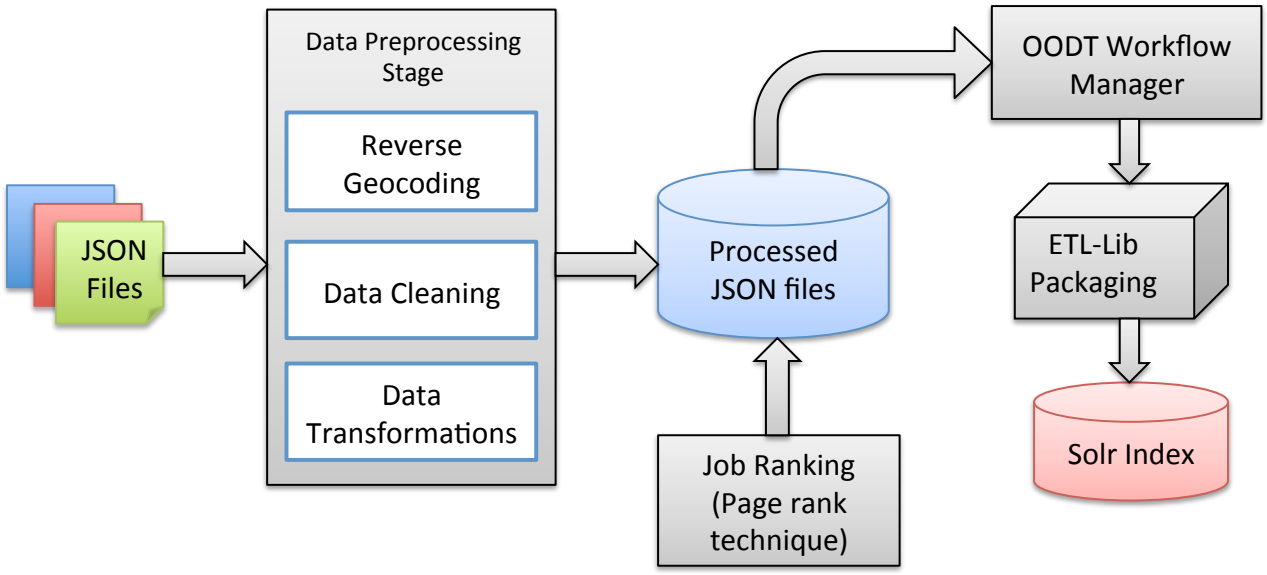

Fig. 1: System block diagram

Mexico		Peru		Colombia	
4137	Auxiliar Contable	2674	Asistente Contable	3703	Auxiliar Contable
3127	Recepcionista	1336	Auxiliar Contable	2046	Asesor Comercial
2916	Auxiliar Administrativo	1148	Secretaria	1478	Recepcionista
1543	Almacenista	993	Asistente Administrativo	1346	Conductor
1505	Ejecutivo de Ventas	991	Recepcionista	1140	Auxiliar contable
1053	Asistente Administrativo	743	Practicante de Contabilidad	1035	Auxiliar de Bodega
1046	Chofer	530	Almacenero	979	Secretaria
980	Contador	528	Asistente de Contabilidad	854	Mensajero
947	Cajera	449	Chofer	783	Asesores Comerciales
930	Ayudante General	447	Asistente Administrativa	738	Asistente Administrativa

Fig. 2: Query 1 results

V. Apache OODT Review

Apache OODT is meant to be a collection of tools that could be easily set up within a few steps. Unfortunately, the time spent to understand the product and its over all integration is large due to the lack of documentation that covers a wider range of examples and use cases. The existing tutorials and wiki are good to get us started using some of the individual components, but does not provide a detailed overall view of how each module could be pipelined.

The dev-mailing list and archives always comes to rescue, but due to the variety of ways these tools could be used, one spends a lot of time in researching and exploring the OODT package in an ad-hoc fashion, wasting time. We had to struggle a lot to get the file concatenator running. Thats because cas-filemgr-¡VERSION¿.jar is not present by default in $WORKFLOW_HOME/lib and $FILEMGR_HOME/lib.

Its given in the instructions that you should for presence, but nothing is given if its not present. We had to figure out, that it needs to be copied from $RESMGR_HOME/lib to both of the above paths. Until we could figure this out, we kep getting following exception: Exception in thread T̈hread-2j̈ava.lang.NoClassDefFoundError: org/apache/oodt/cas/filemgr/metadata/CoreMetKeys

VI. Ranking algorithms

A. Implementation of Content Based Algorithm

In order to improve the job relevancy, we added additional fields to the json that will help in fetching relevant results for a given query. By adding the address info - street, city, state and country, we could group result counts on more structured data fields rather than using the noisy Location field that came with the original json. For content based algorithm we are grouping results based on some fields in the json document which we have identified on per query basis. To get a better subgrouping we have used faceted search query in Solr to drill down results on a finer granularity like Location-¿jobtype-¿postedDate etc. We have implemented function query score to sort the results and show results with a higher score above in the results.

VII. Ranking algorithms

A. Implementation of Link Based Algorithm

For link based algorithm, we identified and derived following links based on following r̈elationshipsb̈etween the each json job file:

1) *JobType:* Job Type can help to link job types more precisely.
2) *City:* City can help in linking job types in the nearby locality.
3) *State:* State can help linking jobs that are present in the same geographical area.
4) *Title:* Since this attribute clearly specifies the relevancy of the job, we have selected it to make links in jobs

We used a python graph framework NetworkX[12] to create the graphs. We connect the jobs when we find similarity any in any of the above attributes. The graph was created in memory and hence we had serious restriction on the amount of nodes we could process. In order to process as much jobs as possible, we created the graph for 15,000 jobs using an Amazon xLarge memory optimized instance.

Once we have the graph created, we ran the pagerank [13] method with damping_factor=0.85, number of iterations = 100 and min_delta=0.00001. The page rank algorithm took 2 hours to compute. We then injected the pagerank field into the json before loading it in SOLR

VIII. Challenge Queries

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

For this query, we take an area name from the user, and identify the top 20 current job types. The examples are shown in the table (Query 1 results) . To get a better subgrouping we have used faceted search query in Solr to drill down results on a finer granularity like Location – jobtype – postedDate etc. We have implemented function query score to sort the results

[12]https://code.google.com/p/python-graph/wiki/Example

[13]https://code.google.com/p/python-graph/source/browse/trunk/core/pygraph/algorithms/
r=724

Mexico		Peru		Colombia	
3631209	Contador General	4015468.28	Asistente Administrativo	2539705.263	Vendedor
3748914	Gerente de Ventas	4190923.28	Practicante de Contabilidad	2607026.087	Auxiliar Administrativo
3760492	Asistente Administrativo	4382276.92	Ingeniero Civil	2819356.098	Asistente Administrativa
3779071	Vendedor	4475983.89	Asistente Administrativa	2829386.982	Ingeniero de Sistemas
3799987	Secretaria	4601106.96	Asistente Contable	2831858.824	Secretaria Auxiliar Contable
3819497	Contador	4687803.35	Asistente contable	2930655.951	Almacenista
4083496	Auxiliar Administrativo	4720168.42	Ayudante de Cocina	2947840	Asesor comercial
4212694	Auxiliar administrativo	4820265.59	Recepcionista	2957493.188	Contador
4233352	Chofer	4824754.49	Auxiliar Contable	2967679.083	Asistente Contable
4241404	Ayudante General	4838015.14	Chofer	2997322.105	Auxiliar contable
4259288	Auxiliar Contable	4881699.31	Personal de Limpieza	3131144.828	Asesores Comerciales
4334582	Ejecutivo de Ventas	4926494.12	Asistente de Gerencia	3150452.093	Servicios Generales
4363542	Mensajero	4990985.66	Almacenero	3173405.279	Asesor Comercial
4516180	Auxiliar de Recursos Human	5087745.36	Asistente de Recursos Humanos	3244856.206	Mensajero
4581761	Auxiliar contable	5101513.59	Secretaria	3249414.637	Auxiliar Contable
4602830	Almacenista	5133281.42	Auxiliar de Contabilidad	3349831.256	Secretaria
4626061	Recepcionista	5203763.05	Practicante Contable	3510112.333	Conductor
4685298	Capturista	5270400	Asistenta Social	3528362.089	Auxiliar de Cocina
5072510	Cajera	5355981.82	Asistente de Contabilidad	3690530.988	Recepcionista
5420360	Auxiliar de Almacen	5445760	Cajera	3836827.826	Auxiliar de Bodega

Fig. 4: Query 2 results

and show results with a higher score above in the results shown in (Query 1 results) .

Hence we can predict that the jobs could further gain popularity. We have implemented a variety of SOLR queries using custom sorting based on average time of a job to get filled. Sorting on average gives better results as compared to raw time of a job to get filled over time. Also, for prediction of results of future job types, it is relevant to show results with areas having higher concentration of jobs higher in the results and hence along with the score, count of jobs in that particular region is used for boosting results to be shown above in the total results found.

B. Compare jobs in terms of quickly theyre filled specifically in regards to region.

In order to answer this question, we first identified the top 20 job types for a given region. Here to get meaningful job types, we used the title field. Once we have the count of the top most occurring job types for a given region, we then find all jobs for having those job types and get the average rate of filling. The avg. time to fill job = sum of difference between the firstSeendate to lastseendate of all jobs. The table (Query

Reception		Sales		assistant		teaching		analyst	
2644	Bogota, D.C.	4444	Bogota, D.C.	33152	Bogota, D.C.	2100	Bogota, D.C.	5841	Buenos Aires City
894	Buenos Aires City	2530	Lima	9398	Lima	563	Cali	5441	Bogota, D.C.
747	Lima	1542	Buenos Aires City	5396	Medellin	294	Puebla	2953	Lima
607	?lvaro Obreg?n	1354	Medellin	5093	Cali	278	Medellin	1325	?lvaro Obreg?n
547	Guadalajara	986	Cali	3336	?lvaro Obreg?n	256	Lima	1064	San Isidro
491	San Luis Río Colorado	904	Guadalajara	3085	Guadalajara	200	Barranquilla	937	San Luis Río Colorado
459	Puebla	845	?lvaro Obreg?n	2737	San Luis Río Colorado	190	Bucaramanga	888	Ate
360	Medellin	814	Puebla	2603	Bucaramanga	142	Iztapalapa	881	Medellin
348	Queretaro	590	Bucaramanga	2564	Barranquilla	134	Buenos Aires City	653	Queretaro
306	Cali	576	Barranquilla	2512	Ate	128	Cartagena	619	Caracas
303	Miguel Hidalgo	559	Queretaro	2313	Puebla	124	Guadalajara	585	Cali
300	Miraflores	554	San Luis Río Colorado	2205	Queretaro	123	Los Olivos	502	Valencia
293	Benito Juárez	536	Merida	2183	Buenos Aires City	117	Chía	456	Miguel Hidalgo
271	San Isidro	491	Monterrey	1975	San Isidro	113	?lvaro Obreg?n	422	Cuauht?moc
249	Naucalpan de Ju?rez	474	León	1815	León	108	Ecatepec de Morelos	406	Monterrey
235	Caracas	395	Cuauht?moc	1694	Callao	101	Miraflores	396	Guadalajara
232	Zapopan	370	Ate	1683	Miraflores	100	Soacha	347	Miraflores
230	León	359	Benito Juárez	1633	Naucalpan de Ju?rez	92	Tlaxcala	321	Puebla
219	Ate	322	Pereira	1568	Miguel Hidalgo	88	Villavicencio	316	Santiago de Surco
219	Benito Ju?rez	306	Iztapalapa	1561	Santiago de Surco	87	San Juan de Lurigancho	302	Cordoba

Fig. 5: Query 3 results

2 results) shows the fastest filling jobs in 3 different regions. The average value being less implies the job fills up faster

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

We identify 5 basic job types and find the density of these job types in different regions. Based on the table (Query 3 results) , we see that Bogota D.C is a very large city and hence has ample opportunities across all fields. But on concentrating at Buenos Aires, we see that most of the Analyst jobs are present there. This indicates that Buenos Aires might have more corporate head quarters as most of the analysts jobs are present there. Secondly, when looking at the teaching field, Cali comes up which indicates a different nature of that region, i.e., having a dense student population and number of educational institutes.

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

We will discuss the trends in our data set with respect to some graphs. As show in the above graph, figure : 6 , the

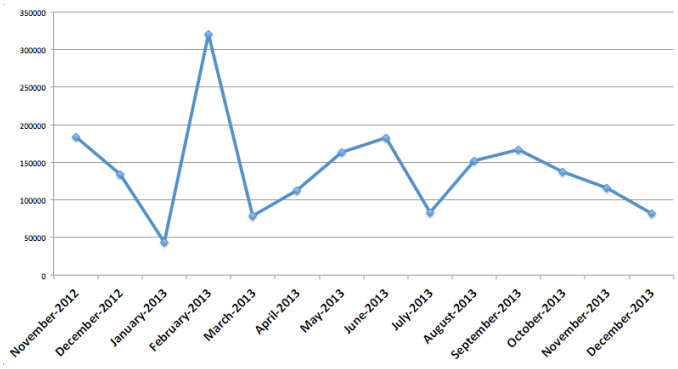

Fig. 6: Total jobs

total number of job openings across the board is shown. We can see that there is a significant hike in recruiting dying the month of feb, which in many countries is nearing the end of the financial year

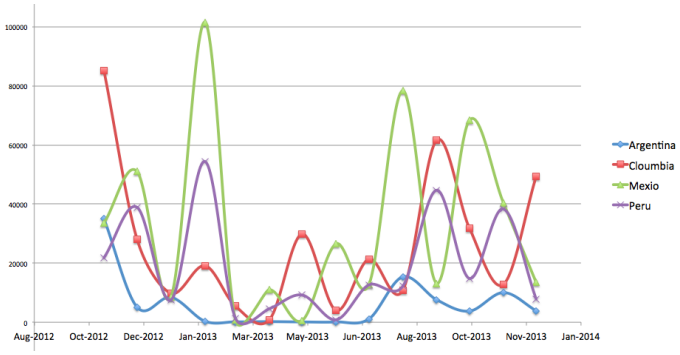

Fig. 7: Over all job lastseendate

The above graphs, figure 7, show shows the number of jobs based on their lastseendate over the period in 4 countries - Mexico, Colombia, Peru and Argentina. As we see Mexico and Colombia had a large recruiting season in the Jan-Feb duration. Their job openings keeps coming up across the years, but when Colombia is compared, we see a different time frame. Colombia has large openings towards the end of 2012 which we eventually filled. The graph below shows the number of job openings based on their firstSeenDate.

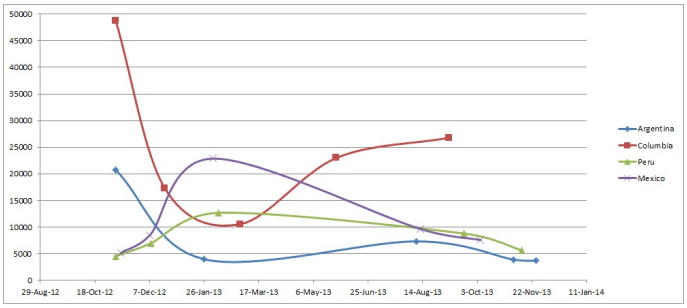

Fig. 8: Full time Job opportunities

For trend analysis, Full Time, Figure:8, jobs have a higher concentration as compared to Part Time, Figure : 9, jobs in South America. To better showcase the results, we drilled down the query results over time on the Posted Date and Last Seen parameter to give out the metrics. A scatter plot with line is plotted using those results to visualize the trends to get a better understanding. Classification of cities based on job data is done based on the highest concentration of jobs for that particular city. Since the data is in Spanish, Spanish to English mapping is maintained to show the results on what classification a city falls under.

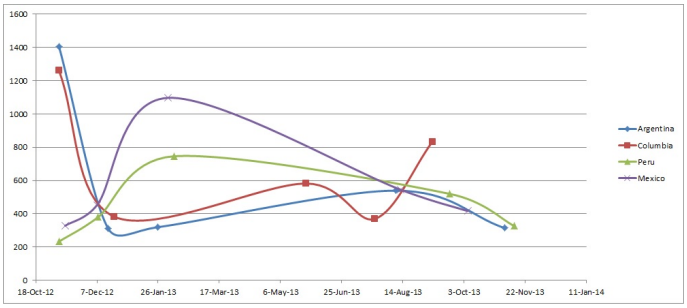

Fig. 9: Part time Job opportunities

IX. Open Source Contribution

1) Communicated over dev-mailing list for all queries.
2) Updated documentation for Apache OODT metadata extractor.