IJCTA, 9(3), 2016, pp. 1679-1685
© International Science Press

Elasticsearch Usage in Automatic Ontology Based Job recommendation system for reliable and accurate notifications

Uma Pavan Kuma K.* **and S. Saraswathi****

Abstract: In this paper we are discussing the automatic ontology construction method for job recommendation system and based on the historical data of the users and their preferences the dynamic notification about the suitable jobs will be posted. The ontology construction we have done automatically and based on the constructed data we are extracting the suitable jobs for the users. The system mainly focuses on generation of the notifications to the users by analyzing the available content in the user context and by taking the job posting forms the job portal context. Here the notifications will be helpful for both job seekers and job portals. Generally employers are spending much time and money for filtering the suitable persons for a job requirement. Similarly job seekers also spending more time for getting the suitable notification related to their skills and experience. Our job recommendation system is a solution for the above problems as we are giving a perfect solution of getting the required information for both job seekers and employers. To implement this we are using slowly changing dimensions along with elastic search. The proposed method will give the notifications to jobseekers and employers along with that we are trying to get reliable job recommendation system so as to eliminate the fake job seekers and fake companies from the notifications.

Keywords: Automatic Ontology, Job recommendation system, Historical data, Preferences, elasticsearch

1. INTRODUCTION

Ontology is a formal explicit specification of shared conceptualization which will provide Common semantics for agent communication. Currently, applications mostly exchange information on the basis of passing parameters or data, formatted according to pre-defined strict syntaxes this approach is known as the exactness method [1]. This method has the advantage of allowing total error management, except application bugs of course, but leaves no space for data interpretation. In consequence, reasoning on data of this type is virtually impossible because of the limits of its definition. Ontologies provide a richer knowledge representation that improves machine interpretation of data. For this they become to be widely used in information systems and applications, and ontology construction has been addressed in several research activities. Ontology must be able to grow dynamically without deviation of the existing applications. At the same time computational time for discovering the best matches between several ontologies is expensive, therefore the technique must maintain previous discovered alignments and common usages in order to quickly recognize similarities between concepts and to compute only new information [2]. Ontology is designed not only to provide a complete view of domain concepts but also to identify quickly and accurately similarities between concepts, even if not identical, and to conduct consistent alignments. The method we are working on gives the answer for the statement that is within the specified domain without taking much time will give the best matching of the job recommendations by considering the previous details and similarities of the key skills or work locations of the job seekers. In the research contribution we are using elastic search which is the emerging technology in the current data processing. The justification of elastic search usage in our research

* PhD Research Scholar, CSE Department, Pondicherry Engineering College, *Email: umapavanmtech@gmail.com*

** Professor & HOD IT Department, Pondicherry Engineering College, *Email: swathi@pec.edu*

is to identify the required information by searching; exactly the elastic search is doing the process of getting the tweets data and Amazon analytics are few cases to mention [3]. One of the main requirements of today's applications is search capability. In the market, we can find a lot of solutions that answer this need, both in commercial as well as the open source world. One of the most used libraries for searching is Apache Lucene. This library is the base of a large number of search solutions such as Apache Solr, Indextank, and ElasticSearch. ElasticSearch is written with both cloud and distributed computing in mind. Thus, the main scope of ElasticSearch is to be a search engine; it also provides a lot of features that allow you to use it as a data store and an analytic engine using aggregations. ElasticSearch contains a lot of innovative features: it is JSON/REST-based, natively distributed in a Map/Reduce approach [4], easy to set up, and extensible with plugins.

In some cases there may be a chance of changing the requirements in the input to the system. In such cases we must use the dynamic ontologies so as to embed the changes in the existing result sets. The organization of the paper is as follows in the section II we are going to explain the method followed in automatic ontology construction for job recommendation system. In section III the details about extension of job recommendation system search, in section IV the explanation about the usage of elastic search for the job recommendation system that we are constructed in section V future scope and conclusion is explained.

2. JOB RECOMMENDATION AUTOMATIC ONTOLOGY (JRAO)

The process we have adopted for the construction of automatic ontology for the job recommendation system is as follows.

Initially, the web log data can be collected from the Job portals. After the raw data can be collected, the pre-processing can be done. Initially, ontology creation and mapping will be done by analyzing various properties of ontologies in order to deduce alternate semantics that may apply to other ontologies, and therefore create a mapping. The feature extraction module based on TF-IDF similarity[5], and then Indexing and ranking of information by Rabin Fingerprint algorithm and ranking can be done by semantic similarity measure. Once the ontology is mapped, the relevant information will be retrieved effectively based on the user query. That is for the input query keyword, matching will be performed with the mapped ontology and the exact information's will be retrieved using the computed similarity score. Lastly, by means of the matching result the related report were generated from the document repository [6]. The implementation is done in Java with protégé software tool for ontology creation and updating. The performance of the proposed system will be analyzed in terms of accuracy, recall and F-measure. In the measuring of the relevancy of the records populated by the .csv files we are using score value of the records generated in such a way that, a threshold value we will mention to get more relevant records with similarity measure. Here the measure will compute the score value out of the existing records and their attributes with the help of word net repository and it outputs the most relevant records with the help of that only the .owl will be generated.

Once the generation of .owl is over the next step is to generate the ontology with set of attributes and the relationships. Initially the entire data items are converted into ontology format and there after we can query the ontology based on the requirements. In the following diagram we can see the attributes and relationships surrounded by ontology and the export of ontograph is possible for the query where workplace is equal to "Bangalore". The following is the sample diagram of Ontograph generated for the query where Skill =”Java Related Development".

The above diagram is the representation of the result in protégé for the query where skill is based on Java and the query is giving results like place of work and description about the jobname through which the user can have a glance of getting the things. Here it is easy to the user to get the instant decision so as to opt for a particular opportunity based on his preferences like skill wise and work location wise. In the next

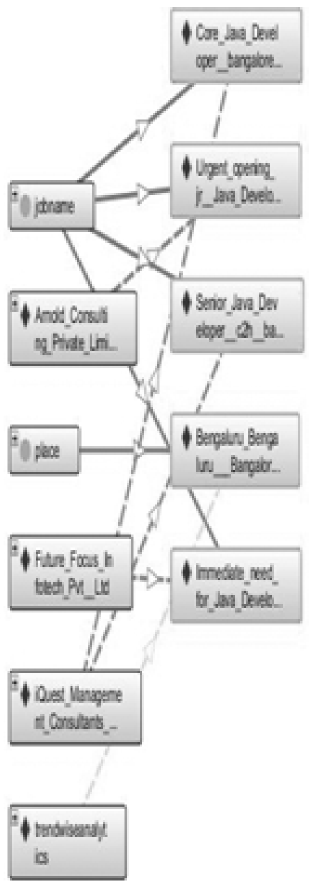
Core_Java_Devel oper__bangalore...
Urgent_opening_ jr__Java_Develo...
jobname
Senior_Java_Dev eloper__c2h__ba...
Arnold_Consulti ng_Private_Limi...
place
Bengaluru_Benga luru__Bangalor...
Future_Focus_In fotech_Pvt__Ltd
Immediate_need_ for_Java_Develo...
iQuest_Manageme nt_Consultants_...
trendwiseanalyt ics

Figure 1: Ontograph diagram for the query Skill = ”Java”.

section we are giving the details about extension of created Job recommendation system in the context of reliability and accuracy of the decision making in the context of jobseekers and employers.

3. EXTENSION OF JRAO

To strengthen the job recommendation automatic ontology we are proposing the embedding of additional information to the extracted records. The process will be in two phases. The phases are classified based on employers and job seekers. The employers will get the notifications about suitable candidates as per his requirements such as qualification and skill kind of queries. Whereas the job seekers will be getting the notifications like the company information and credibility and work culture in that company all these things will be available in the notifications of the jobseekers.

The following discussion explains the detailed activities in each phase.

i) Phase-I: In the context employee the job seekers details he expects is whether the said details are real or at least up to the minimum measures. To materialize it our proposal is to get the data from LinkedIn, academia and other professional membership sites and compare the details of the populated records with the extracted information. Here we will get some outcome like similarity of the data or else contradiction or mismatched kind of the results. From this it is easy to conclude the profile of a particular person. Based on the results of the comparison this phase will suggest the employers about a jobseeker.

ii) Phase-II: In the context of Jobseekers even they want to know the credibility of the company like how they will pay and promotion policies and any other benefits to the employees. To achieve this we are getting the review about the company and getting the opinion of previous employees and these things will be helpful to conclude about the company. Based on the outcome the notifications can be sending to the jobseekers so that he can decide about his association with the company.

To implement the above phases usage of tf and idf alone are not enough so we are using the elasticsearch along with job recommendation system.

4. ELASTICSEARCH USAGE

Elasticsearch is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine. Accessible through an extensive and elaborate API, Elasticsearch can power extremely fast searches that support our data discovery applications [7,8].

4.1. Benefits of Using Elasticsearch:

On the same hardware, queries that would take more than 10 seconds using SQL will return results in under 10 milliseconds in Elasticsearch. A query examines one or many target values, and scores each of the elements in the results according to how close they match the focus of the query [9]. The query operators enable you to optimize simple or complex queries that often return results from large datasets in just a few milliseconds. The Elasticsearch design is much simpler and much leaner than a database constrained by schemas, tables, fields, rows, and columns.

Elasticsearch converts raw data such as log files or message files into internal documents and stores them in a basic data structure similar to a JSON object. Each document is a simple set of correlating *keys* and*values:* the keys are strings, and the values are one of numerous data types—strings, numbers, dates, or lists. Adding documents to Elasticsearch is easy—and it's easy to automate.

Full text searches will be extremely fast because the documents are stored in close proximity to the corresponding metadata in the index [10]. This design greatly reduces the number of data reads, and ES limits the index growth rate by keeping it compressed.

4.2. Working Model Of Elasticsearch

Elastic Search is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead. This type of index is called an **inverted index**, because it inverts a page-centric data structure (page $\rightarrow$ words) to a keyword-centric data structure (word $\rightarrow$ pages). Elastic Search uses Apache Lucene to create and manage this inverted index. By default, **Elasticsearch** makes use of the **Lucene's** practical scoring formula, which represents the relevance score of each document with a positive floating-point number known as the _score**.** The higher the _score the higher the relevance of the document [11]. A query clause generates a_score for each document, and the calculation for that score depends on the type of query clause. Query clauses serve different purposes: a fuzzy query might determine the **_score**by calculating how similar the spelling of the found word is to the original search term; a terms query would incorporate the percentage of terms that were found. However, the more common meaning of relevance is the algorithm that calculates the similarity of the contents of a full-text field in comparison to a full-text query string [12,13].

The standard similarity algorithm used in **Elasticsearch** is known as term frequency/inverse document frequency, or **tf/idf**, and it takes the following factors into account:

Term Frequency (tf) is a measure of the number of occurrences of a term in a document in context. If the occurrence count is high, the score will be high, and the chances for inclusion of that document as relevant will be high.

Inverse Document Frequency (idf) is a measurement of how frequently the search terms occur across a set of documents. Typically, if the search term commonly occurs across many documents, the score will be low. Frequent occurrence of rare words will typically boost the score value.

Coord is a measurement of matching on multiple search terms, and a higher value of this measurement will increase the overall score. Consider a search for the two terms, "Hadoop" and "Map Reduce." It doesn't matter if you issue a term query: this will run internally as a **bool** query and separate searches will run for each of the terms. A document that has both of these words will get a higher rank than documents containing either of the search terms.

lengthnorm measures smaller field matches and gives these more weight. For example, if the search term finds a match in a title field instead of the content field, it may achieve relevance. Although it is not directly related to document relevance, a **querynorm** is a measure for comparing queries when you are using a combination of query types.You can also affect the score with either of the boost factors **index time boost** and **query time boost**. Boosting a specific field can cause it to have more significance in the score calculation.

$$score(q, d) = query\ Norm(q) * cord(q, d)$$
$$* \sum (tf(tind) * idf(t)^2$$
$$* t.\ get\ Boost(\ ) * norm(t, d))\ \ (t\ ind)$$

5. PERFORMANCE IMPROVEMENT OF JRAO WITH ELASTICSEARCH

In the design of Automatic ontology we have used the score values to compute the relevancy in terms of term frequency (tf) and Inverse Document Frequency (idf) [14, 15]. If we could use elasticsearch, the performance will be improved based on the following criterion.

- Searching of large volumes of data
- Indexing Documents to the Repository
- Direct access to the Data
- Broadly Distributable and Highly Scalable
- Multiple search items score calculation
- Label based searching
- Complex query composition with JSON
- Fastest query processing time compared with schema based storage.

5.1. Simple Querying With JSON

The JSON usage is preferable in elasticsearch querying, and which is simple even in the processing of nested or complex queries [16].

In the following representation of JSON [16] we can post the information of jobseeker who is having his Job_id=20.

```
curl -XPOST 'localhost:9200/Job_Ontology/_search?pretty′ -d′
        {
                        ″query″: {″match″: {″Job_Id″: 20}}
         }
```

The following is the query is getting the information about the jobs posted between given dates.

```
Curl-XGET localhost:9200/Job_Ontology/_search?pretty′ -d
{
″query″:{ range″:{ ″postDate″:{″from″:″2015-12-10″,″to″:″2016-02-29″}  } }
}
```

The representation is simple and powerful, even we can use Ruby/Python scripting so as to implement the elasticsearch [17].

6. RESULT ANALYSIS

The results of our experiment are analyzed in two stages. In the initial stage the normal searching process for getting the suitable employee by the employers. Here the procedure is simple and straight forward like manually opening the website surfing the site by giving the required keywords in the form of qualification, skill and location etc., But in our implementation just we will give the web sites as inputs and the corresponding .csv's will be generating and from that the .owl will generated. The ontology will use the generated .owl and we can observe the automatic construction of ontology along with relationships and other querying options. The generated ontology can be exported as ontograph and .DOT formats for the purpose of reuse. Whereas the later stage focuses on extension of the search criterion like to identify more relevancies from the short listed data. Here we are focusing on getting the support of professional bodies for the authentication and validity of the data populated by the ontology. The following is the table which gives the overview of the results we are focusing on. In this table we have embedded the parameters which are involved in initial and later stages. The significance of this analysis is to re verify the populated information from the source data and is a 2-stage validation check for the job seekers and employers.

Table 1:2
Stage Job Recommendation Input, operations and output

	I/P		*Operations*		*O/P*
STAGE1	Webpages	.Csv Generation	.Owl Generation	Used tf and idf	Automatic ontology generation
STAGE2	Populated Data	Professional networks association	Score vale generation Using Elasticsearch	Used tf, idf, coord, lengthroom, querynorm	Notifications

The above table gives a brief description of various inputs, operations and output at a 2-stage process. In stage1 the input is web pages and we applied the logic to extract the information of job name, skills, company name and location kind of the data, these things are stored into the format of the .csv file. Based on the input number of webpages those many .csv's will be created. Next operation is generation of the .owl for the given query for this we have used term frequency and Inverse Document frequency as measures. Based on the .owl we are generated the automatic ontology with various attributes and relationships.

The stage 2 is meant for getting the score value of the records which will decide the quality of the data in the context of reality of the data. To implement this we are using the generated records as input and as the operational aspects integration of the input value with professional network data like LinkedIn, the purpose is getting the score value as per the available records with respect to the identified professional network data values. For getting this score value we dependent on elasticserach which evaluates term frequency, inverse document frequency, coord, length room and query norm. Based on the above measures we can generate the notifications to job seekers and employers.

7. CONCLUSION

The current article describes the construction of the Job recommendation system with automatic ontology. Once the construction is over we will be getting the relevant job notifications based on the score values given by the system with similarity measure. But we want to get more authenticated information by getting the details from other professional web data like LinkedIn. At this stage the outcome is either supportive or ambiguous, based on this result the jobseekers or employers can take the decisions. One more thing we have done in the score generation is usage of elasticsearch in the score value generation so that more relevancies we can get in the processing of the huge volumes of the data. The ultimate goal of this research is to establish a job recommendation system with the support of automatic ontology construction after that getting the more relevant recommendation to both jobseekers and employers so as to get accurate and reliable information from the populated data. The job recommendation system initially developed by automatic ontology base and in the next stage the job recommendation system is embedding the n so as to get the reliable information about the job seekers and employers. In the next stages the usage of elasticsearch in the job recommendation system will give the accurate result set based on the various score values. The job recommendation system we are expecting more accurate and reliable when compared with manual job processing system and other benefits are time saving quality and systematic.

References

[1] "Nelia Lasierra, "Designing an Architecture for Monitoring Patients at Home: Ontologies and Web services for Clinical and Technical Management Integration," IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 18, NO. 3, MAY 2014"p. 456-464.

[2] Mark Roantree, "Automating the Integration of Clinical Studies into Medical Ontologies", 2014 47th Hawaii International Conference on System Science.

[3] Vishal Jain, "Architecture Model for Communication between Multi Agent Systems with Ontology", International No. 8, May-June 2013, pp. 234-244.

[4] Pavel A. Smirnov, "Domain Ontologies Integration for Virtual Modelling and Simulation Environments", Procedia Computer Science, Elsevier, Volume 29, 2014, pp. 2507–2514.

[5] S. MONISHA, "A FRAMEWORK FOR ONTOLOGY BASED LINK ANALYSISFOR WEB MINING", Journal of Theoretical and Applied Information Technology, 20th March 2015. Vol. 73, No. 2 pp. 120-128.

[6] Vishal Jain, "MINING IN ONTOLOGY WITH MULTI AGENT SYSTEM IN SEMANTIC WEB: A NOVEL APPROACH", The International Journal of Multimedia & Its applications (IJMA) Vol. 6, No. 5, October 2014, pp. 45-55.

[7] Fernández López, M., "Overview Of Methodologies For Building Ontologies", Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 2013.

[8] Pratibha Gokhale, "Ontology Development Methods", DESIDOC Journal of Library & Information Technology, Vol. 31, No. 2, March 2011, pp. 77-83.

[9] Ying Ding, "Ontology Research and Development –A Review of Ontology Mapping and Evolving", Journal of Information Science, 28(5), 375-388, 2012 pp. 12-24.

[10] Pragya Gupta, "Survey paper on elasticsearch", International Journal of Science and Researc(IJSR), ISSN:2319-7604, Volume 5, Issue 1, November 2015, pp. 333-337.

[11] Oleksii Kononenko, Mining modern repositories with elasticsearch, MSR'14, May 31–June 1, 2014, Hyderabad, India.

[12] Helena Sofia Pinto, "A Methodology for Ontology Integration", K-CAP'01, October 22-23, 2001, Victoria, British Columbia, Canada, Copyright 2001 ACM 1-58113-380-4.

[13] Trong Hai Duong, "Complexity Analysis of Ontology Integration methodologies: a Comparative Study", Journal of Universal Computer Science, vol. 15, no. 4 (2009), pp. 120-134.

[14] Mark Roantree, "Automating the Integration of Clinical Studies into Medical Ontologies", IEEE Computer Society, 2014.

[15] Yuxia Huang, "Using Ontologies and formal concept analysis to integrate heterogeneous tourism information", IEEE Transactions in Emerging Topics in Computing, June 2015.

[16] O. Iroju, "State-of-the Art: A Comparative Analysis of Ontology Matching Systems", African Journal of Computing & ICT, Vol. 5, No. 4, June 2012, IEEE ISSN 2006-1781.

[17] Yuxia Huang and Lingbian Using Ontologies and formal concept analysis to integrate heterogeneous tourism information, IEEE Transactions in Emerging Topics in Computing, June 2015.