Spring 2018

ONTOLOGY BASED TECHNICAL SKILL SIMILARITY

Yeshwanth Bashyam Balachander
San Jose State University

ONTOLOGY BASED TECHNICAL SKILL SIMILARITY

A Writing Project

Presented to

The Faculty of Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Computer Science

By

Yeshwanth Bashyam Balachander

May 2018

SAN JOSÉ STATE UNIVERSITY

The Undersigned Thesis Committee Approves the Thesis Titled

Ontology Based Technical Skill Similarity

By
Yeshwanth Bashyam Balachander

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Teng Moh, Department of Computer Science 5/9/2018

Dr. Robert Chun, Department of Computer Science 5/9/2018

Prof. James Casaletto, Department of Computer Science 5/9/2018

Abstract

Online job boards have become a major platform for technical talent procurement and job search. These job portals have given rise to challenging matching and search problems. The core matching or search happens between technical skills of the job requirements and the candidate's profile or keywords. The extensive list of technical skills and its polyonymous nature makes it less effective to perform a direct keyword matching. This results in substandard job matching or search results which misses out a closely matching candidate on account of it not having the exact skills. It is important to use a semantic similarity measure between skills to improve the relevance of the results. This paper proposes a semantic similarity measure between technical skills using a knowledge based approach. The approach builds an ontology using DBpedia and uses it to derive a similarity score. Feature based ontology similarity measures are used to derive a similarity score between two skills. The ontology also helps in resolving a base skill from its multiple representations. The paper discusses implementation of custom ontology, similarity measuring system and performance of the system in comparing technical skills. The proposed approach performs better than the Resumatcher system in finding the similarity between skills.

Keywords—Semantic similarity, Job recommender systems, Job search, Ontology based similarity.

Acknowledgement

I am grateful to my project advisor Dr. Teng Moh for his guidance and insights throughout the project that helped in steering it in the right direction. I would also like to thank my committee members Dr. Robert Chun and Professor James Caseletto for their support and time to the project.

I would like to thank San Jose State University and Computer Science department for this opportunity and the resources to pursue this project. Finally, I would like to thank my family for their continued support and encouragement all through my Masters journey.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

In the Internet age, the number of services provided online have grown exponentially. One of the important and widely used service is online job boards. In these job boards, keyword search has been the main channel for job seekers to search for relevant job postings. This creates a setup where technical skills are compared exactly and their similarities are ignored in the matching process. Adding to it, the continuous growth in the number of technical skills and its multiple representations have made the direct match process less effective in getting relevant content. Addressing the “problem of relevancy”, intelligent recommendation systems are built to recommend jobs that match a candidate’s profile. Content Based (CB) techniques are predominant in job recommendations [1] and most of these approaches also resort to a similarity measure based on direct matching of skills.

Direct matching approach fails to consider the relationship between various skills to determine a similarity measure. This will result in related skills not being matched between the resume and job description. For example, "Mobile Development" and "Android" being related skills in a semantic sense, becomes dissimilar in the direct matching approach. Another example will be the skill - "Java Database Connectivity" not being matched with "JDBC" in a direct matching approach. Despite being the same skill the direct matching fails to understand the different representations of the same skill.

The aim of this work is to improve the relevancy of technical skill matching using a knowledge based semantic similarity approach. A custom skills ontology is built by crawling DBpedia [2] and used as a knowledge base to understand the relationship between skills [3]. Feature based semantic similarity measures are used to derive a similarity score between skills in the ontology. Apart from understanding the relationship between skills, the approach also aims at understanding the multiple representations of the same skill in deriving the score.

The paper is organized as follows. Section II surveys the existing works related to ontology based similarity measures and similarity measures used in job recommendations. Section III explains the details of the proposed approach and Section IV gives a brief overview of the system architecture. Details of the implementation of the proposed approach is discussed in Section V. Section VI summarizes the evaluation results of the system's performance and Section VII concludes the paper with pointers to potential future improvements.

2 Related Works

The section surveys existing approaches in the categories of ontology based similarity measures [4] in general and skill similarity measures used in job recommendation systems.

2.1 Ontology Based Similarity Measures

Ontology based similarity measures can be broadly categorized as follows

a. Edge counting approaches

b. Feature based approaches

c. Information Content based approaches.

This survey assesses the first two categories for its pros and cons in applying it to the custom ontology. Considering ontology as a directed graph of concepts, each concept is connected to other concepts through taxonomic relationships. Edge-counting approaches use these relationship paths to determine the similarity score. The taxonomic relationship in the skills ontology will be the *specialization* relationship (subclass) between concepts. For example, skill "NoSQL" and "MongoDB" will be connected with a "subclass" relationship, representing the fact that the latter concept is a specialized version of the former. Edge-counting approaches considers the path between two concepts in the ontology to arrive at a similarity score. Path can be defined as $path\ (a,b) = e\_1, e\_2, e\_3\ \ldots e\_n$ where e_i represents the edges connecting the two concepts in the ontology. The work in [5] proposes an edge-counting approach, which considers the length of the minimum path between two concepts as the measure of dissimilarity. Concepts far away in the ontology will have a high dissimilarity score as compared to concepts in close proximity.

$$dis(a,b) = \min_{\forall i} |path_i(a,b)| \qquad (1)$$

The approach has a simple intuition but assumes that the minimum path truly reflects the similarity of two concepts. This might not work in the ontology built from DBpedia as there could be several paths between two concepts based on trivial categories. For example, DBpedia categorizes "Python" and "Visual Basic" under "Programming languages created in 1991". Despite having other meaningful categories for these skills, the edge counting approach will result in a minimum distance of 1. This makes these two skills highly similar while they are not related.

Another drawback of the edge counting approach is that it considers all category levels to have the same weightage. More specialized categories (higher depth categories) should contribute more to the similarity than general (shallow depth) categories. Wu & Palmer [6] proposed an approach considering the depth of the concept in the ontology. The similarity identifies the Least Common Subsumer (LCS) of the two skills and uses the distance to the LCS - N1, N2 and depth of the LCS - N3 to arrive at a score.

$$sim(A,B) = \frac{2*N3}{N1+N2+2*N3} \qquad (2)$$

The major drawback of edge-counting approaches is that it considers every path in the ontology to have uniform weight or distance [4]. Feature based approaches overcome this by considering the ontological features of concepts and its overlap to determine similarity. The features of a concept are determined by the nature of the ontology. The ontology built in this approach is taxonomically oriented as *specializations* and hence the ancestors of a concept act as its features.

Rodríguez and Egenhofer [7] computes similarity using the ratio of common features to the total number of features. The total features are discounted using a factor γ(a,b) which is a function of the depth of a and b. (3) gives the similarity measure formula

$$sim(a,b) = \frac{|A \cap B|}{\gamma(a,b)|A \setminus B| + (1-\gamma(a,b))|A \setminus B| + |A \cap B|} \qquad (3)$$

$$\gamma(a,b) = \begin{cases} \dfrac{depth(a)}{depth(a)+depth(b)} & depth(a) \leq depth(b) \\ \dfrac{depth(b)}{depth(a)+depth(b)} & depth(b) \leq depth(a) \end{cases}$$

Sanchez et al. [8] proposes a feature based dissimilarity method using the ratio of dissimilar features of the two concepts to the total features. The similarity applies non-linearity using log_2 to limit the result in the range of [0 1]. Features of a skill is defined as Ø(a)=ancestors(a). Dissimilarity between two concepts are defined in (4)

$$dis(a,b) = \log_2(1 + \frac{|\emptyset(a) \setminus \emptyset(b)| + |\emptyset(b) \setminus \emptyset(a)|}{|\emptyset(a) \setminus \emptyset(b)| + |\emptyset(b) \setminus \emptyset(a)| + |\emptyset(b) \cap \emptyset(a)|} \qquad (4)$$

$$|\emptyset(a) \setminus \emptyset(b)| : Length\ of\ features\ of\ 'a'\ not\ in\ features\ of\ 'b'$$

$$|\emptyset(a) \cap \emptyset(b)| : Length\ of\ features\ common\ to\ both\ 'a'\ and\ 'b'$$

As the proposed implementation uses an ontology based on specialization relationship, ancestor-feature based methods appear to be an ideal choice. In this method, the features of each skill will be immediate generalization categories of the skill. Calculating similarity or dissimilarity between these features can provide a true sense of similarity between two skills.

2.2 Similarity Measures in Job Recommendation

Job recommendation systems, in general follow similarity strategies based on direct matching. Feature vectors are generated for job descriptions and resumes. The similarity between these feature vectors are derived using a distance measure such as Cosine distance, Jaccard distance, Pearson Correlation etc. These features do not consider the various representations of the skill or its relationship to other skills.

The iHR+ reciprocal job recommendation system [9] arrives at a feature vector which combines several attributes such as educational degree, skills, location, work status etc. Their approach uses Cosine Similarity in calculating similarity between the two vectors. The work in taxonomy based job recommendation [10] considers a taxonomy based approach in building the feature vector of the candidate. The system uses the O*NET database [11] of occupation and skills to include relevant skills into the job seeker's profile vector. For example, when the job seeker's profile contains the skill- "Java Developer", the system extracts related skills from the O*NET database and adds them into the job seeker's feature vector. This is done in a view to improve the matching between job description and a user's profile.

S. Guo et al. work in Resumatcher [12] uses a combination of ontology and Jaccard Similarity to calculate the similarity between skills. Their approach uses ontology to identify if two skills are directly related as hypernyms or hyponyms. If the skills are related, the similarity score is calculated as the ratio of the number of documents containing both the skills to the total number of document containing either of the skills.

3 Proposed Approach

The proposed approach is to build a custom ontology using DBpedia data and derive the similarity measure using feature based measures such as Sanchez et al. [10] and Rodriguez et al. [9] feature based similarity. Ontologies have proven to be a successful tool for measuring semantic similarity in challenging natural language problems. In Information Science, Ontology of a domain contains information about the formal naming and definition of types, properties and relationship between entities in that specific domain [13]. Several semantic similarity applications and research use lexical databases such as WordNet [14], DBpedia [2] and Yago [15].

These ontologies apply well for generic applications but have drawbacks with domain specific applications. The availability of a standard ontology for technical skills is limited resulting in the need for a custom-built ontology. Consequently, the lack of domain specific ontology makes knowledge based approaches less effective, leading applications to use data mining approaches to identify semantic similarity. However, the success of data mining approach is dictated by the quantity and the quality of training data. Additionally, data mining approaches derive similarity based on the co-occurrence of skills in a large corpus of data. This might not always judge the real similarity of two skills based on its properties. For example, Data mining related approaches might term “Javascript” and “HTML” as similar skills as they tend to co-occur in many job descriptions. However, it is well known that these two are totally different type of languages. It is necessary to build a custom ontology, specific to the technical skills domain to overcome these shortcomings.

DBpedia is chosen as the source of information to build the ontology. It is a project aimed at extracting structured content from the information available in Wikipedia. The vast number of technical skills and their fast-evolving nature imposed a challenge on selecting the right source of information for this task. Since DBpedia is extracted from Wikipedia, the information available covers a wide range of resources in the domain of software technology. Moreover, DBpedia provides querying capabilities to semantically query relationship between Wikipedia resources. Although DBpedia is an ontology in itself, it is not directly used in the proposed approach. Content related to technical skills are systematically crawled from DBpedia and a custom ontology with the required taxonomy is built. This helps reducing the information scope and filtering out resources and properties which are of less importance to the problem at hand. This is also done in a view to optimize performance during the actual similarity calculation.

4 System Architecture

The system architecture involves three major components – DBpedia crawler, also referred to as the Ontology builder, Technical Skills Ontology (TSO) and Skill Similarity. The architecture of the system is depicted in Fig 1.

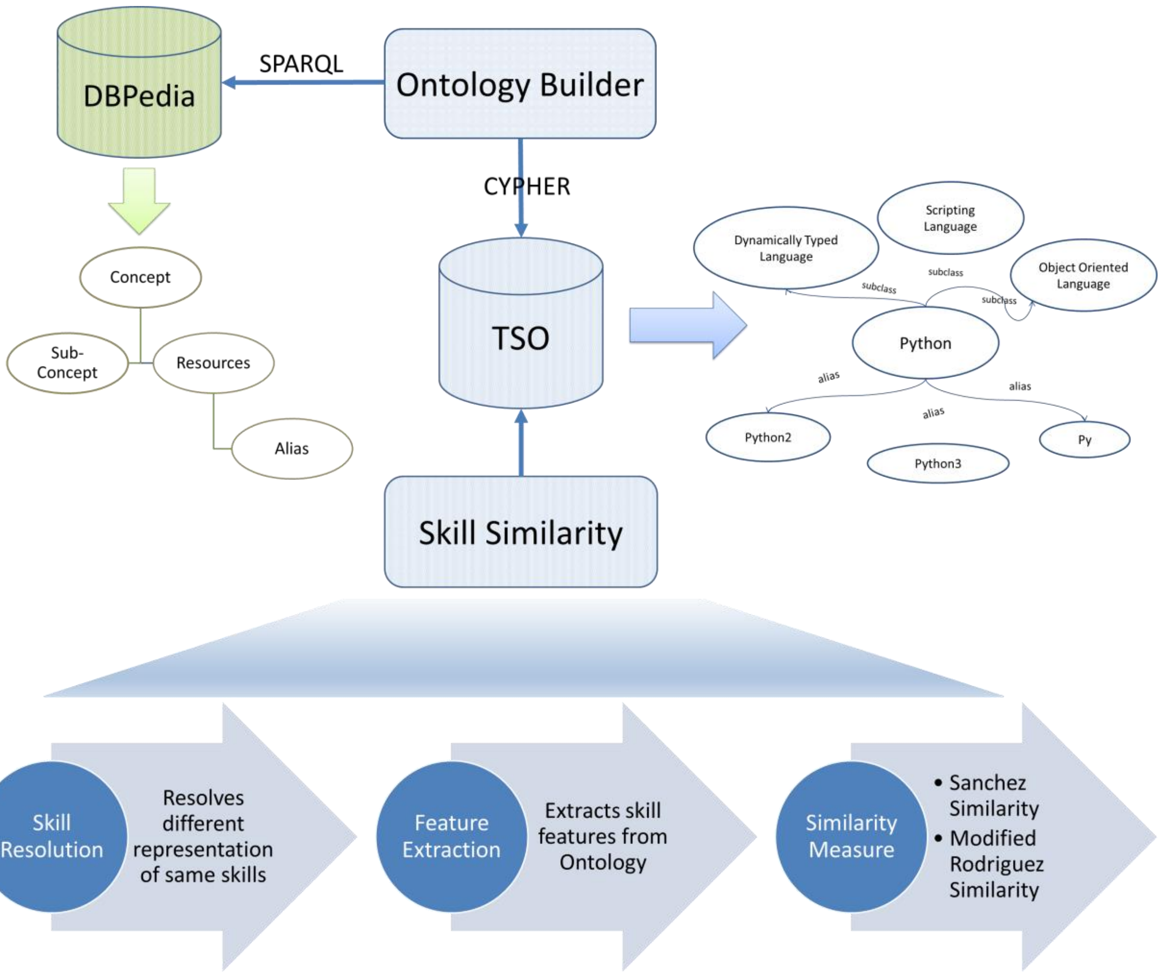

Fig. 1. System Architecture

The DBpedia crawler is the component responsible for systematically crawling relevant information from DBpedia. The component contains the algorithm to crawl from a seed set of DBpedia resources and persist the relationship between the resources into the custom TSO. The component plays a major role in filtering the mammoth DBpedia information into the custom ontology, specific to the job domain. The component takes care of filtering out content irrelevant to technical skills and transforming it into a consistent taxonomical structure.

The quality of a knowledge based system relies on the quality of the knowledge base itself. In this case, the custom ontology serves as the knowledge base and hence the quality of the system heavily relies on the ontology crawler component which builds this ontology.

The TSO component is the custom built technical skills ontology. The system is built as a persistent store of the ontology with its concepts and taxonomical relationships. Each node in the TSO represents either one of “technical skills”, “technical skill categories” or “skill aliases”. The technical skill nodes are the nodes corresponding to an actual skill. The skill category nodes represent generalization categories of a set of technical skill nodes. The skill alias nodes represent the multiple representations of a single technical skill node. The relationship between these nodes represent one of ‘subclass’ or ‘alias’ relationship. The ‘subclass’ relationship marks the specialization line between two nodes. The ‘alias’ relationship marks the possible aliases a skill possesses. The structure of the possible nodes and their relationship is depicted in Fig 2. The TSO component provides capabilities of relationship based querying on the taxonomy of technical skills. Neo4j [16] – a graph based database is used as the persistent storage for the ontology. Neo4j provides performance optimized graph based queries which will be used by the feature based Skill Similarity component.

Skill Similarity component is the core component deriving similarity between technical skills. The component interacts with the TSO in extracting features of technical skills and applies the feature based similarity measure. The first step of the component is skill resolution, where a skill name is resolved to a base skill. The resolved base skill will be a “technical skill” node in the ontology. The process of resolving the base skill involves the utilization of the “skill alias” node and the "alias" relationship. The second stage extracts the features for the base skills using the custom ontology. In the proposed approach, the ancestors of a skill acts as its features. The intuition driving this is that, two skills which are similar, share common ancestors in the ontology. The "subclass" relationship is utilized to extract the ancestors of a skill. The ancestors fetched by the feature extraction stage are fed into the last stage which calculates the similarity score. Two feature based similarity measures are applied and a score is returned for the two skills in comparison. The scores are limited between 0 to 1 and can be viewed as the probability of similarity.

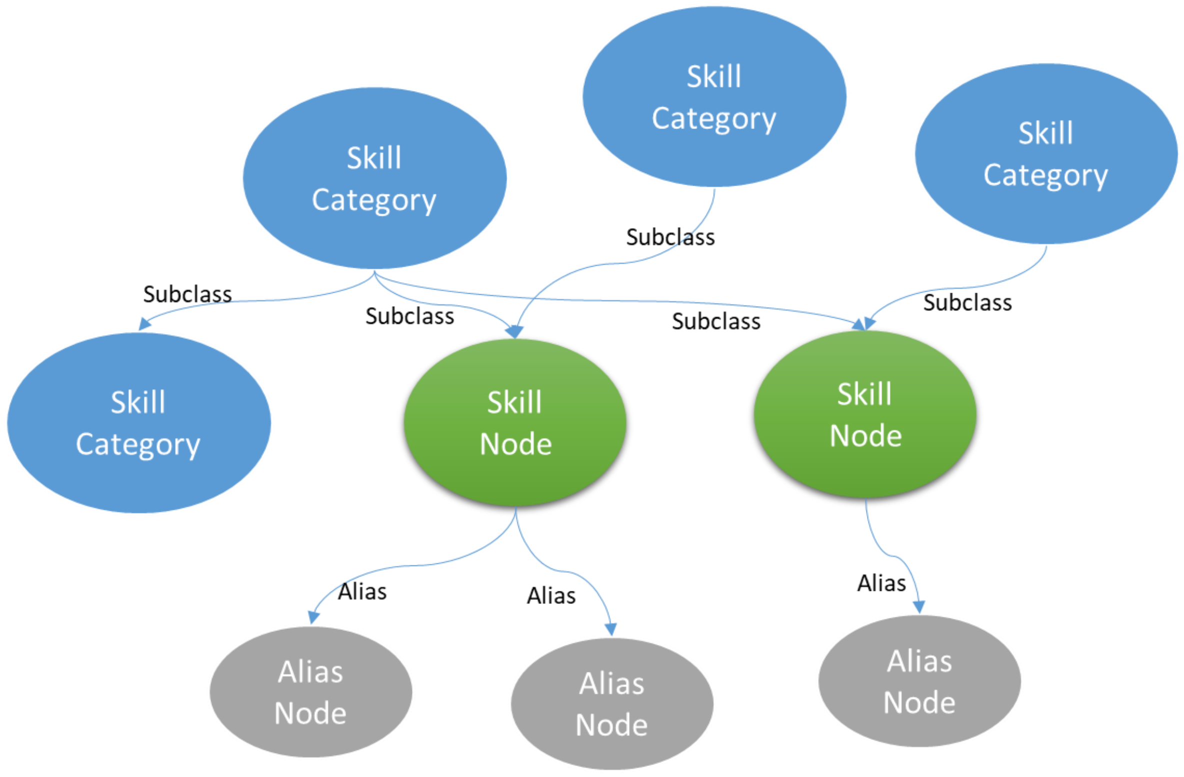

Fig. 2. TSO Taxonomy

5 Implementation

The section covers the implementation of the various components required for the proposed approach.

5.1 DBpedia

A project aimed at extracting structured content from the information available in Wikipedia, is used as the information source to build the ontology.

DBpedia [2] is built based on the information in Wikipedia making it meet the requirements of the skills ontology. Additionally, DBpedia contains information categorized as entity classes and concepts. This information can be queried for relationship and properties using an SQL like query language called the SPARQL (SPARQL protocol for RDF query language). SPARQL is a query language used for querying structured data stored in the Resource Description Framework (RDF) format. The four major DBpedia elements that are used in building the custom ontology are Concepts, Resources, Type and Wiki Page Redirects. Each of these elements allow us to understand more about the dbpedia entity that is being queried. This can be used to guide the crawler in building the custom ontology. The Concepts in DBpedia represents categories of information and it directly translates to “Skill Categories” in the custom ontology. A Concept can have multiple sub-concepts associated with it and a Concept can be the subject of several Resources. For example, “Computer Programming” is a DBpedia Concept which has around 30 sub-concepts including “Computer libraries”, “Programming Languages” etc.

The Resource in DBpedia directly translates to the “Skill Node” in the custom ontology. A Resource can be associated with multiple Concepts. For example, the Resource “Python” has around 5 associated Concepts including “Dynamically typed programming language”, “Programming Language”, “Object oriented programming language” etc. The Type attribute in DBpedia plays a major role in helping to filter relevant content. Each Resource has one or more Types associated and the crawler looks for specific types to extract the technical skills. The current implementation looks for two dbpedia types – “Software” and “Programming Language”. The crawler also looks for Yago types which are integrated with DBpedia. Finally, the “Wiki Page Redirects” attributes are translated to “Alias” nodes in the custom ontology. These attributes provide the different representation of a skill and a DBPedia resource can have more than one “Wiki Page Redirect” attributes.

The sample DBpedia information about the entity “Databases” is illustrated in the Fig 3. The root concept contains sub-concepts such as “NoSQL”, “Relational”, “Online Database” etc. Each of these concepts contain a set of resources associated. The associated resources of “NoSQL” are “MongoDB”, “CouchDB”, “DocumentDB" etc.

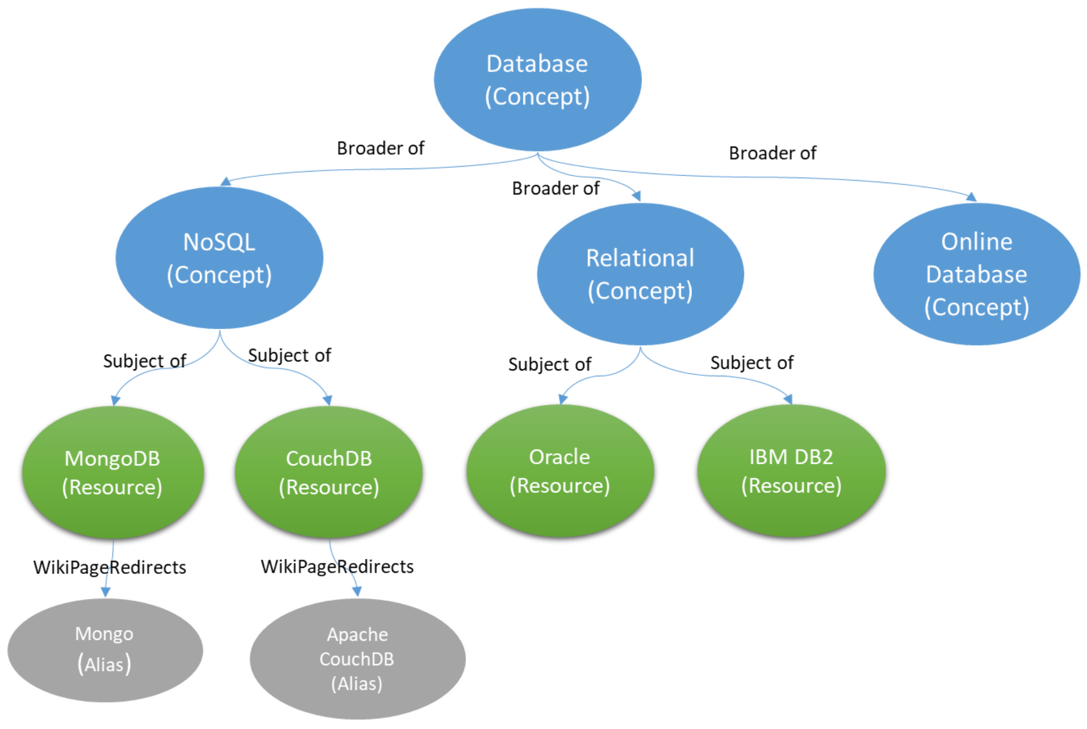

Fig. 3. DBpedia taxonomy structure

5.2 DBpedia Crawler

The crawler component is implemented to build the custom ontology by crawling the information from DBpedia. DBpedia uses terminologies such as concept and resources which are analogous to categories and resources respectively.

The crawler performs a Breadth First Search (BFS) on the DBpedia ontology graph by starting from a base set of skills. These base skills are identified by a unique URI. While processing a concept, the crawler fetches all the DBpedia resources that contains the concept as its subject using the ‘subject’ attribute. Of all these resources, the ones which are of type ‘Software’ or ‘Programming Language’ are filtered out. These two are the corresponding DBpedia types given for software or programming languages which are our resources of interest in this problem. These filtered resources are added as children to the concept currently being processed. Additionally, the crawler fetches all the sub-concepts using the’ broader’ attribute of each concept. These sub concepts are added to the initial seed set and the algorithm continues till no more concept in the seed set is available for processing. Algorithm 1 describes the ontology builder algorithm.

TABLE I. RESOURCE QUERY

```
Select distinct ?resource ?resource_label where{{
 ?resource <http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Web_development>.
 ?resource <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type.
 ?resource <http://xmlns.com/foaf/0.1/name> ?resource_label
 FILTER
(?type=<http://dbpedia.org/ontology/Software>
||?type=<http://dbpedia.org/ontology/Language>)
}}
```

The crawler is implemented in python to perform the BFS and to trigger the required SPARQL queries. The BFS is limited by a depth parameter to prevent it from crawling into deeper levels of the DBpedia data. The implementation uses SPARQLWrapper and RDFLib libraries to interface with DBpedia APIs.

Algorithm 1 **Custom Ontology Builder**

```
1: procedure BUILD ONTOLOGY($seedSet$)
2:     $processQueue \leftarrow seedSet$
3:     while $processQueue \neq \{\}$ do
4:         $resource \leftarrow processQueue.head$
5:         $subResources \leftarrow dbpedia.$GET_SUBRESOURCES(resource)
6:         if LEN(subResources) $\geq 1$ then
7:             tso.MERGE_SKILLS($subResources$)
8:             tso.MERGE_RELATIONSHIP($resource, subResources,$ “$subclass$“)
9:             $subConcepts \leftarrow dbpedia.$GET_SUBCONCEPTS(resource)
10:            if LEN(subConcepts) $\geq 1$ then
11:                tso.MERGE_SKILLS($subConcepts$)
12:                tso.MERGE_RELATIONSHIP($resource, subConcepts,$ “$subclass$“)
13:            $processQueue \leftarrow processQueue \cup subConcepts$
14:    return
```

Table I and II provide the sample SPARQL queries for the “Web development” concept. Table I contains a sample resource query which fetches all the DBpedia resources with its subject as “Web development”. Table II contains the sample sub-concept query which fetches all the DBpedia sub concepts for “Web development”.

TABLE II. SUB CONCEPT QUERY

```
PREFIX broader: <http://www.w3.org/2004/02/skos/core#broader>
select distinct ?sub_concepts ?concept_label where{{
    ?sub_concepts broader:
<http://dbpedia.org/resource/Category:Web_development>.
    ?sub_concepts <http://www.w3.org/2000/01/rdf-schema#label>
?concept_label.
    }}
```

As DBpedia is based on content from Wikipedia, it is important to handle junk data. It is also important to clean the name of the entities and concepts fetched and transform it to a consistent representation. The crawler algorithm is designed to take these factors into consideration. The crawler adds concepts to the ontology graph only when it has at-least one resource which is either a software or a programming language. This prevents junk concepts from being added to the ontology database. The crawler understands the naming representation in DBpedia entities and cleans them from delimiting special characters before adding to the TSO.

5.3 TSO

The ontology that is being built –TSO, is stored in a graph based database called Neo4j. Neo4j supports storage and retrieval of data in a graph based format which could contain nodes and edges. The database supports a graph based query language called Cypher [16] which is optimized for faster retrieval and updates of graph data.

As the crawler fetches technical skills and its dependent skills, it stores them in the Neo4j database. Each skill is stored as a node and a directed edge between two skills indicates a specialization relationship. The additional advantage of storing the ontology in Neo4j is that it makes querying the relationship between skills easier and faster. Furthermore, the query language also supports in-built graph functions such as shortest path algorithms. Cypher queries are used to find relationship between two skills.

TABLE III. ANCESTORS QUERY

```
MATCH (p:Skill)<-[:subclass*..2]-(c:Skill)
WHERE c.name =~'(?i)kotlin' RETURN p.name as name
```

The flexibility of the query language allows to find the relationship of two skills at various levels. The main use case for the approach is to find the ancestors of a given skill. The other use case for the approach is to resolve a base skill from the alias. Table III shows the sample query that fetches the ancestors of the skill - “Kotlin”.

5.4 Skill Similarity

Feature based similarity measures are used to derive the similarity score between two skills in the TSO. In this approach, the ontology's taxonomical hierarchy is used as the feature set to understand the similarity between two nodes in the ontology. The fundamental ideology of this approach is that the semantic similarity of two concepts can determined using the common concepts that subsume it. When two concepts in the ontology are similar, they tend to have more common ancestors subsuming it. Since TSO is built based on specialization and generalization taxonomy of technical skills, the approach concurs well for skill similarity. This can be visualized through Fig 1 where "MongoDB" and "CouchDB" are similar skills which share the same generalization of "NoSQL". Since the two skills share more common ancestors, their similarity score is evaluated higher than two nodes not sharing many ancestors.

The proposed approach assesses two different feature based similarity metrics. Sanchez Dissimilarity [8], discussed in (5) determines the dissimilarity of two concepts based on the ratio of dissimilar features to the total features of the compared concepts. The approach applies non-linearity using log2 which also helps in bounding the result in the range of [0,1]. The result can also be viewed as a probability of how dissimilar two skills are. As a result, we could derive the similarity as $1 - dis_{sanchez}(a,b)$

$$dis_{sanchez}(a,b) = log_2\left(1 + \frac{|\emptyset(a)\backslash\emptyset(b)|+|\emptyset(b)\backslash\emptyset(a)|}{|\emptyset(a)\backslash\emptyset(b)|+|\emptyset(b)\backslash\emptyset(a)|+|\emptyset(a)\cap\emptyset(b)|}\right) \qquad (5)$$

The second similarity measure used is modified from the Rodriguez similarity [7] discussed in (3). This approach derives similarity based on the ratio of similar features to the ratio of dissimilar features. The TSO contains many generalization concepts extracted from Wikipedia. Few of these ancestor concepts could have little importance in representing the actual skill and can overpower when considering the total count. To help in balancing this problem, the total number of features are truncated based on the density of the feature space. The parameter gamma helps in achieving the truncating effect. This can approximately help the similarity measure counter the noise caused by less important features. Similar to [10], Log2 non-linearity is applied to bound the similarity values between [0,1]. The resulting modified Rodriguez equation (6) is the second measure used by the approach

$$S_r(a,b) = \log_2\left(1 + \frac{|\emptyset(a) \cap \emptyset(b)|}{|\emptyset(a) \cap \emptyset(b)| + \gamma(a,b)|\emptyset(a) \setminus \emptyset(b)| + \left(1 - \gamma(a,b)\right)|\emptyset(b) \setminus \emptyset(a)|}\right) \qquad (6)$$

$$\gamma(a,b) = \begin{cases} \dfrac{|\emptyset(a)|}{|\emptyset(a)| + |\emptyset(b)|} & |\emptyset(a)| \leq |\emptyset(b|) \\ \dfrac{|\emptyset(b)|}{|\emptyset(a)| + |\emptyset(b)|} & |\emptyset(b)| \leq |\emptyset(a)| \end{cases}$$

6 Experiments and Results

Evaluating a semantic similarity measure is a challenge as the perception of similarity between two concepts differ from one person to another. This subjectivity is a result of varied cultural, geographical and domain background. An unbiased and controlled survey of similarity scores for humans are imperative to evaluate the performance of any semantic similarity. Similarity measures between English words are evaluated against standard benchmarks created by curating surveys from humans in a controlled environment [18]. Some of the common benchmarks used for English languages are RG65 dataset established by Rubenstein and Goodnough and WS353 dataset. RG65 contains 65-word pair scores judged by 51 humans. WS353 dataset contains 353 word pairs judged by 13 to 16 humans.

Semantic similarity between technical skills being a unique problem, there are no standard benchmarks to compare the proposed approach against. However, the work in Resumatcher [12] had performed a human evaluation of skills similarity using 12 domain experts for 10 unique skills. This provides similarity score for 55 distinct skill pairs. Table IV summarizes the pairwise scores for the 55 skill pairs.

The proposed system uses two similarity measures and a configurable parameter for the number of ancestors features to consider. The pairwise scores generated by the proposed system using Rodriguez sim1ilarity measure is summarized in Table V

TABLE IV. PAIRWISE SCORES BY DOMAIN EXPERTS

Skill	***JS***	***C#***	***PHP***	***CSS***	***Java***	***Python***	***Ruby***	***Bash***	***SQL***	***C++***
JS	1									
C#	0.6	1								
PHP	0.63	0.58	1							
CSS	0.58	0.3	0.48	1						
Java	0.62	0.8	0.58	0.3	1					
Python	0.64	0.58	0.56	0.36	0.7	1				
Ruby	0.7	0.48	0.63	0.49	0.58	0.64	1			
Bash	0.51	0.42	0.54	0.35	0.44	0.49	0.5	1		
SQL	0.38	0.3	0.4	0.42	0.33	0.4	0.4	0.3	1	
C++	0.5	0.82	0.53	0.29	0.84	0.64	0.56	0.42	0.31	1

Adopting these scores in Table IV as the benchmark, the performance of the proposed approach is evaluated using Normalized Discounted Cumulative Gain (NDCG). NDCG (8) is an evaluation metric to determine the quality of content retrieved in information retrieval problems. The evaluation considers the positions of the content in the result list in determining the quality of the information retrieval. The evaluation works with the assumption that highly relevant documents at the top of the result leads to higher quality in information retrieval.

TABLE V. PAIRWISE SCORES BY PROPOSED SYSTEM

Skill	***JS***	***C#***	***PHP***	***CSS***	***Java***	***Python***	***Ruby***	***Bash***	***SQL***	***C++***
JS	1.00									
C#	0.36	1.00								
PHP	0.51	0.14	1.00							
CSS	0.26	0.00	0.09	1.00						
Java	0.21	0.29	0.30	0.15	1.00					
Python	0.60	0.16	0.59	0.16	0.31	1.00				

Ruby	0.60	0.16	0.55	0.16	0.31	0.79	1.00			
Bash	0.46	0.19	0.36	0.19	0.12	0.58	0.58	1.00		
SQL	0.34	0.32	0.17	0.32	0.22	0.25	0.36	0.32	1.00	
C++	0.35	0.47	0.24	0.17	0.41	0.26	0.26	0.33	0.39	1.00

The equation for the Discounted Cumulative Gain(DCG) is given in (7). As the similarity scores could be different from one approach to another, it is essential to normalize the scores to compare it against another system. The DCG values are normalized with Ideal Discounted Cumulative Gain (IDCG). IDCG is the DCG for the perfect ordering (human ordering) of the results.

$$\text{DCG: } \sum_{i=1}^{p} \frac{2^{rel_i - 1}}{log_2(i+1)} \quad (7)$$

$$NDCG = \frac{DCG}{IDCG} \quad (8)$$

Ordering skills based on its similarity scores with other skills gives a result ordering similar to information retrieval. Considering the ordering determined by the human scores as the ground truth, NDCG can be applied to measure the quality of similarity for the various skills orderings. Table VI provides the ordering of skills related to "Java" based on the scores in Table V in comparison with ordering based on scores in Table IV. Higher similarity scores result in the skill appearing on top of the list. The ordering being more similar to the human ordering results in a higher NDCG score for the corresponding skill. It can be observed from the table that the system ordering matches closely with the human ordering for the top 3 skills.

TABLE VI. SKILLS ORDERING COMPARISON FOR JAVA

Skill	**Human Score**	**Skill**	**System Score**
Java	1	Java	1.00
C++	0.84	C++	0.46

C#	0.8	Python	0.32
Python	0.7	Ruby	0.32
JS	0.62	PHP	0.30
PHP	0.58	C#	0.29
Ruby	0.58	SQL	0.23
Bash	0.44	JS	0.21
SQL	0.33	CSS	0.15
CSS	0.3	Bash	0.15

The NDCG scores of the 10 skills are compared against the NDCG scores of the Resumatcher system. Table VII summarizes the Resumatcher score with μ=0.96 and σ=0.04.

TABLE VII. RESUMATCHER NDCG SCORES

Skill	**JS**	**C#**	**PHP**	**CSS**	**Java**	**Python**	**Ruby**	**Bash**	**SQL**	**C++**
NDCG	0.96	0.905	0.988	0.993	0.898	0.985	0.981	0.99	0.987	0.951

The experiment is conducted using two similarity measures – Sanchez (5) and modified Rodriguez similarity (6) with a hyper parameter controlling the number of ancestor hops to consider.

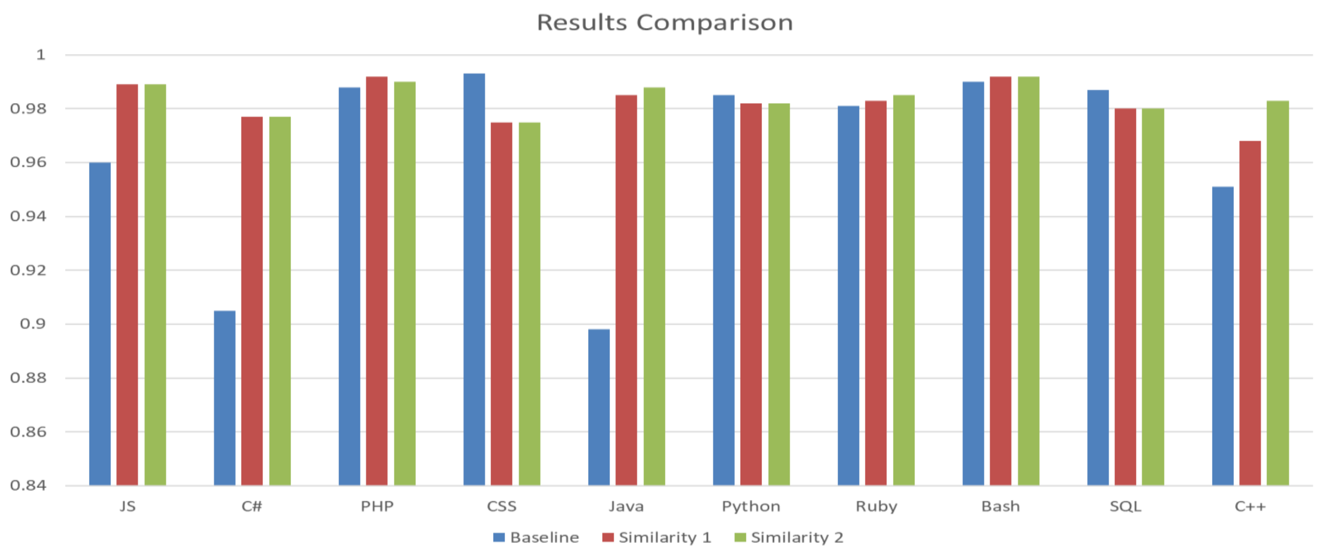

Fig. 4. NDCG Scores Comparison

Total of 4 different pairwise scores are generated and NDCG is calculated using these scores for all 10 skills. Table VIII summarizes the results of the experiment. The results show that the proposed system performs consistently better than the

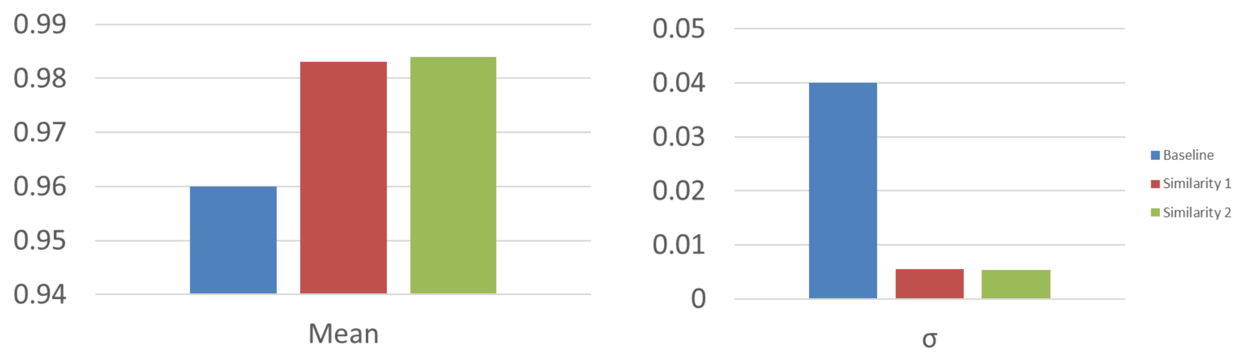

Fig. 5. Mean and Variance Comparison

TABLE VIII. PROPOSED SYSTEM NDCG SCORES

Skill	Resumatcher	Sanchez		Modified Rodriguez	
		3 Hops	***2 Hops***	***3 Hops***	***2 Hops***
JS	0.96	0.988	0.989	0.988	0.989
C#	0.905	0.976	0.977	0.964	0.977
PHP	0.988	0.992	0.992	0.988	0.99
CSS	0.993	0.972	0.975	0.967	0.975
Java	0.898	0.989	0.985	0.991	0.988
Python	0.985	0.98	0.982	0.98	0.982
Ruby	0.981	0.986	0.983	0.986	0.985
Bash	0.99	0.991	0.992	0.987	0.992
SQL	0.987	0.974	0.98	0.974	0.98
C++	0.951	0.982	0.968	0.982	0.983

μ	**0.96**	**0.983**	**0.983**	**0.979**	**0.984**
σ	**0.04**	**0.007**	**0.0055**	**0.01**	**0.0054**

Resumatcher system by the metrics of mean and variance. Comparing the scores for the individual skills, the proposed system has better scores for 8 skills. These scores are better by a notable margin when compared to Resumatcher. Even the two skills for which the system scores are within a negligible range of less than ~ 0.02. The proposed system has a much lower variance highlighting the consistency of the scoring. Out of the 4 pairwise scoring, both modified Rodriguez with 2 hops and Sanchez with 2 hops provided the best scores.

In a view to assess the performance of the system against the performance of data-mining based semantic similarity approaches, an experiment was conducted to compare similarity scores of Point-wise Mutual Information (PMI) [18]. The basis of datamining based approaches to semantic similarity is the assumption that highly similar words tend to co-occur. PMI is one such method based on the above assumption which is a measure of association of two concepts in a large corpus. PMI is calculated using the formula in (9), which is the ratio of probability of the two concepts occurring together, over the individual probabilities of the two concepts.

$$PMI\ (a, b) = \frac{p(a,b)}{p(a)p(b)} \qquad (9)$$

Using the Dice job dataset [19] of 22K technical job description, PMI for the 55 pairs of skills were calculated. The probability of a skill is calculated as the number of documents containing the skills over the total number of documents. The documents were indexed using ElasticSearch and Lucene queries were used to find the required skill counts.

The resulting NDCG of the PMI approach is compared with the best result of the proposed approach. The proposes system had a higher mean NDCG with a very low variance.

TABLE IX. PMI AND PROPOSED SYSTEM NDCG

Skill	***PMI***	***Proposed System***
JavaScript	0.987	0.989
C#	0.969	0.977
PHP	0.98	0.99
CSS	0.992	0.975
Java	0.947	0.988
Python	0.976	0.982
Ruby	0.979	0.985
Bash	0.987	0.992
SQL	0.982	0.98
C++	0.951	0.983
μ	**0.975**	**0.984**
σ	**0.014**	**0.0054**

Adding to it, the quality of data-mining related approaches depend on the quality of training data while knowledge based system only depends on the quality of the ontology. This will avoid cold start problems in recommendations.

7 Conclusion

As stated in the introductory sections, semantic similarity between technical skills could be explored to achieve high relevance in job recommendations and job based search results. Existing systems perform a direct matching between skills to determine similarity which fails to consider the relationship between skills into the similarity process.

The paper proposes an ontology based similarity measure for technical skills using the taxonomy of technical skills. The paper also discusses the implementation of the custom ontology built by crawling DBPedia. By evaluating the different types of ontology based similarity measures, the approach uses feature based methods to derive the similarity score. The results of the system are compared against human evaluated scores captured in [12]. The system consistently performs well against the results of [12] and provides a higher mean NDCG score than the Resumatcher system. The system is also compared against an alternate approach on semantic similarity – PMI and proved to be performing better. The system possesses the advantage of adapting to new set of skills and understand the various representations of the same skill. Additionally, the proposed approach possesses an advantage of containing a comprehensive set of technical skills and its categories though the TSO. This is achieved by the use of DBpedia as a data source for the TSO. The knowledge base enables the system to perform better under cold-start conditions without having the need for training data.

The quality of the similarity in the proposed approach is mainly based on the quality of the custom ontology. The quality of the ontology can be improved by minimizing irrelevant concepts and maximizing the relevant concepts. The proposed system uses the ontology types to identify a technical skill. This can be further improved by fine tuning the ontology crawler to filter using more specialized types. As a future work, the proposed similarity measure can be used in an existing job recommendation system to understand the improvements at a recommendation level. The current system focusses on technical skill and the TSO is built accordingly. To extend the system to more generalized skills, the ontology crawler can be updated to filter the corresponding DBpedia resources. This can be achieved by identifying the correct DBpedia types for the generic skills domain and including them in the crawler. The remainder of the system can be used as is on the updated ontology to derive similarity scores for skill in the generic domain.

References

[1] N. D. Almalis, G. A. Tsihrintzis, and N. Karagiannis, “A content based approach for recommending personnel for job positions,” IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications, 2014.

[2] Mendes, P. N., Jakob, M., & Bizer, C. (2012, May). DBpedia: A Multilingual Cross-domain Knowledge Base. In LREC (pp. 1813-1817).

[3] A. Doan, J. Madhavan, P. Domingos, A. Halevy, "Ontology matching: A machine learning approach" in Handbook on ontologies, Springer, pp. 385-403, 2004

[4] S. Harispe, D. Sánchez, S. Ranwez, S. Janaqi, and J. Montmain, “A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain,” Journal of Biomedical Informatics, vol. 48, pp. 38–53, 2014.

[5] R. Rada, H. Mili, E. Bicknell, and M. Blettner, “Development and application of a metric on semantic nets,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 1, pp. 17–30, 1989.

[6] Z. Wu and M. Palmer, “Verbs semantics and lexical selection,” Proceedings of the 32nd annual meeting on Association for Computational Linguistics -, 1994.

[7] M. Rodriguez and M. Egenhofer, “Determining semantic similarity among entity classes from different ontologies,” IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 2, pp. 442–456, 2003.

[8] D. Sánchez, M. Batet, D. Isern, and A. Valls, “Ontology-based semantic similarity: A new feature-based approach,” Expert Systems with Applications, vol. 39, no. 9, pp. 7718–7728, 2012.

[9] Wenxing, H., Yiwei, C., Jianwei, Q., & Yin, H. (2015). IHR+: A mobile reciprocal job recommender system. 2015 10th International Conference on Computer Science & Education (ICCSE). doi:10.1109/iccse.2015.7250296

[10] M. Diaby and E. Viennet, "Taxonomy-based job recommender systems on Facebook and LinkedIn profiles," 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014.Taxonomay based approach

[11] "O*NET-SOC Taxonomy," O*NET Resource Center. [Online]. Available: http://www.onetcenter.org/taxonomy.html. [Accessed: 04-Apr-2018].

[12] Guo, S., Alamudun, F., & Hammond, T. (2016). RésuMatcher: A personalized résumé-job matching system. Expert Systems with Applications, 60, 169-182. doi:10.1016/j.eswa.2016.04.013

[13] Ontology (information science). (2018, January 23). Retrieved January 24, 2018, from https://en.wikipedia.org/wiki/Ontology_(information_science)

[14] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An On-line Lexical Database*," International Journal of Lexicography, vol. 3, no. 4, pp. 235–244, 1990.

[15] Farzaneh Mahdisoltani, Joanna Biega, Fabian M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. CIDR, Jan 2013, Asilomar, United States.

[16] Webber, Jim. "A Programmatic Introduction to Neo4j." Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity - SPLASH 12, 2012, doi:10.1145/2384716.2384777.

[17] Lofi, C. (2015). Measuring semantic similarity and relatedness with distributional and knowledge-based approaches. Information and Media Technologies, 10(3), 493-501.

[18] Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. Proceedings of GSCL, 31-40.

[19] "US jobs on Dice.com - dataset by promptcloud," data.world, 18-Sep-2017. [Online]. Available: https://data.world/promptcloud/us-jobs-on-dice-com. [Accessed: 04-Apr-2018].