Article

Ontology-Based Personalized Job Recommendation Framework for Migrants and Refugees

Dimos Ntioudis [1,*], Panagiota Masa [1], Anastasios Karakostas [2], Georgios Meditskos [1,3], Stefanos Vrochidis [1] and Ioannis Kompatsiaris [1]

[1] Centre for Research & Technology Hellas, Information Technologies Institute, 6th Km Charilaou—Thermi, 57001 Thessaloniki, Greece
[2] Draxis Environmental, 54655 Thessaloniki, Greece
[3] School of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
* Correspondence: ntdimos@iti.gr

Abstract: Participation in the labor market is seen as the most important factor favoring long-term integration of migrants and refugees into society. This paper describes the job recommendation framework of the Integration of Migrants MatchER SErvice (IMMERSE). The proposed framework acts as a matching tool that enables the contexts of individual migrants and refugees, including their expectations, languages, educational background, previous job experience and skills, to be captured in the ontology and facilitate their matching with the job opportunities available in their host country. Profile information and job listings are processed in real time in the back-end, and matches are revealed in the front-end. Moreover, the matching tool considers the activity of the users on the platform to provide recommendations based on the similarity among existing jobs that they already showed interest in and new jobs posted on the platform. Finally, the framework takes into account the location of the users to rank the results and only shows the most relevant location-based recommendations.

Keywords: recommendation systems; job matching; ontologies; reasoning; migrants; refugees

check for updates

Citation: Ntioudis, D.; Masa, P.; Karakostas, A.; Meditskos, G.; Vrochidis, S.; Kompatsiaris, I. Ontology-Based Personalized Job Recommendation Framework for Migrants and Refugees. *Big Data Cogn. Comput.* **2022**, *6*, 120. https://doi.org/10.3390/bdcc6040120

Academic Editors: Konstantinos Kotis and Dimitris Spiliotopoulos

Received: 29 July 2022
Accepted: 8 October 2022
Published: 19 October 2022

1. Introduction

The issue of whether migrants and refugees are able to contribute to the economic growth and success of their hosts has become an increasingly hot topic. While it is widely accepted that labour mobility contributes positively to economies and ultimately welfare standards and reduces unemployment, it is not a given that migrants are afforded the same opportunity to find employment, become self-employed, finance their own living standards and contribute positively to their hosts' economies [1]. Participation in the labour market is seen as the most important factor favouring long-term integration into society. In service of this goal, Information and Communication Technology (ICT) tools play a vital role in the integration and social inclusion of migrants and refugees [2–5]. They can be used to support migrants in several activities and to overcome the difficulties they might encounter when they come in the destination country, ranging from learning the language of the new country, acquiring job-related skills, accessing education and job opportunities, assimilating with the wider community, and so on.

In this paper, we present the job recommendation framework of the IMMERSE platform. IMMERSE is a web-based platform that was designed and developed in the context of a European project called ICT Enabled Public Services for Migration (MIICT) [6]. The proposed framework acts as a matching tool that enables the contexts of individual migrants and refugees, including their resume and preferences to facilitate their matching with the job opportunities available in their host country. An overview of the architecture and the core components of IMMERSE was presented in [7]. The main objective of IMMERSE is to encourage migrants to overcome significant difficulties they face when they arrive in the host country. This is achieved by offering a variety of services and information that can help migrants integrate with the wider population. For example, migrants can use an IMMERSE platform for finding employment, learning a new language, finding courses to improve their skills, obtaining volunteering assistance and finding social activities. In addition, a preliminary version of the ontology-based personalized job recommendation framework was presented in [8].

Here, we significantly extend our previous approach by: (a) updating the ontology to also capture information about the skills of a user, the job preferences of the user as well as the interaction of the user with existing posts, and (b) extending the matchmaking service with new rules that consider the aforementioned additions in the ontology. The main contributions of this paper can be summarised as follows: (a) the final version of the knowledge representation model of IMMERSE that is capable of semantically representing the Curriculum Vitae (CV) of a job seeker details and the details of a job posting and (b) an ontology-based matchmaking service that provides relevant job recommendations considering the full CV of a job seeker and the details of the available jobs.

The rest of the paper is organized as follows: In Section 2, we describe related work. In Section 3, we present a brief overview of the IMMERSE architecture and the main technical components. Section 4 describes the IMMERSE Ontology and the semantic representation of basic notions. In Section 5, the Semantic Reasoning is presented. Section 6 demonstrates the matchmaking mechanism through a simulation example. Section 7 presents the evaluation results. Finally, Section 8 concludes the paper.

2. Background and Related Work

Ontologies are important in the Semantic Web because they provide a mechanism to deal with heterogeneous representations of web resources. They are a formal way of representing knowledge within a topic of interest, consisting of a range of ideas, their attributes, and the relationships between them. Ontologies can be used to specify various classes, properties, and relationships, as well as rules, axioms, and restrictions. In addition, to derive implicit knowledge hidden into metadata and discover inferences based on asserted information in the ontology, reasoning systems are used. Reasoners are systems that can handle and apply the semantics and can be used to make logical inferences. In particular, ***rule-based*** reasoning is used to create new information in a controlled way, based on very specific and clearly defined rules. The following subsections provide a brief overview of existing ontologies and standards that will form the basis of the IMMERSE ontology and reasoning.

2.1. Knowledge Representation

The use of ontologies requires an ontology language that is well-designed, well-defined, and web-compatible, as well as supporting reasoning tools. The Resource Description Framework (RDF) is a simple knowledge representation language that aims to standardize metadata and descriptions of Web-based resources definition and use [9]. The basic building block in RDF is a set of ***subject-predicate-object*** triples. Moreover, the Web Ontology Language (OWL) is a set of knowledge representation languages that is commonly used to create ontologies and was created to display rich and more complex information about things, groups of things, and relationships between them [10]. OWL has become the official World Wide Web Consortium (W3C) recommendation [11] for authoring and sharing ontologies and extends existing Web standards for representing knowledge such as the RDF, XML, etc.

2.2. User Profile Ontologies

Friend of a Friend (FOAF) [12] is an ontology that captures the fundamental characteristics of a person, e.g., name, gender, date of birth, location and aims to describe people as well as the relationships that connect people, places, and other objects. Similarly, a set of ontologies called the Core Vocabularies [13] encapsulate the core properties of an entity, such as a person or a government agency. The Core Vocabularies were created in response to the requirement for semantic interoperability in the context of European public services, as well as the lack of universally agreed data models and other semantic interoperability-related disputes. These vocabularies are context-neutral data models that are simple, reusable, and expandable.

2.3. Ontologies Relevant to E-Recruitment, Work Experience and Jobs

Job Description ontology [14] is used to describe the semantic structure for defining a job position and to provide a common ground for sharing knowledge. It specifies details about the job title, requirements, responsibilities, education, post date, last-apply date, organization name, etc. Description of a Career (DOAC) [15] is a vocabulary for describing a worker's professional skills. In addition, Ontology for Skill and Competency Management [16] facilitates the management of available human resources' skills and competencies. This ontology assists managers in knowing the competencies of available human resources and matching them with the existing requirements. ResumeRDF [17] is a Semantic Web ontology for expressing information contained in a personal resume or Curriculum Vitae (CV). This includes a greater number of properties and covers details such as work and academic experience, skills, certifications, courses attended and other relevant information. Finally, the Human Resources Management Ontology [18] describes the details of a job posting and the CV of a job seeker by acting as a common "language" in the form of a set of controlled vocabularies. This ontology is composed of thirteen modular ontologies: Competence, Compensation, Driving License, Economic Activity, Education, Geography, Job Offer, Job Seeker, Labour Regulatory, Language, Occupation, Skill and Time.

2.4. Rule Languages

Simple Protocol and RDF Query Language (SPARQL) [19] is a declarative language for extracting and updating information from RDF graphs that is recommended by the W3C community. It is an expressive language that allows the creation of complex relations between different entities and contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. Apart from that, SPARQL can be used to describe rules for generating new RDF graphs by combining data from existing graphs into new ones and by using the CONSTRUCT graph pattern. CONSTRUCT determines which RDF triples should be added to the new graph based on a set of matching criteria applied to existing graphs (i.e., WHERE clause). The CONSTRUCT query form returns a single RDF graph defined by a graph template. The output is an RDF graph generated by substituting the variables in the graph template for each query solution in the solution sequence, then setting union to join the triples into a single RDF graph.

2.5. Recommendation Systems for Job Recruitment

Recommendation systems are frequently utilized to help users find products or services that best fit their individual preferences. Thus, a personalized recommendation system is a solution to the problem of information overload and widely applied in many domains. For this reason, many studies have been conducted and are presented in this section. Firstly, an approach to the Semantic Matchmaking for Job Recruitment is presented in [20]. The aim of the Semantic Matchmaking for Job Recruitment approach is to present a hybrid ontology-based strategy for effectively matching job seekers with job advertisements. For each job posting, they distinguish between nice-to-have skills and must-have skills and also define the level of proficiency for each competence. Similarly, the matchmaking framework proposed in this work considers both nice-to-have requirements and must-have requirements as well as different educational levels and language competency levels to appropriately rank candidates that equally match an open position and improve the final recommendations.

Another solution in the same direction is a recommendation approach in [21] that is recommending jobs to candidates based on their preference profiles which are in turn based on previous preference ratings. Our method also takes into account the preferences of the migrants, i.e., posts that users have marked as favorites or applied to in the past, to favor posts that are more similar during the final ranking. Two stages are carried out to determine whether two jobs are similar: (a) first, we assess the requirements, including the required language, education, and working conditions, and (b) we use text-based similarity algorithms to compare the titles and descriptions of two given jobs. A final similarity score is then computed by combining the results of the two phases.

In addition, Ref. [22] presents Proactive, which is a comprehensive job recommendation system that assists job seekers in a variety of ways in finding appropriate opening jobs. The system depends on a series of ontologies for collecting and storing job related metadata such as the job's category, required educational level, experience and position type (i.e., full time, part time, etc.). This set of metadata fields is also extended with geographic information. This allows for calculating distance-based similarity between users' geographical preferences and job locations. In order to provide jobs that fit a user's profile and are nearby them, our matchmaker additionally uses geographical information from both job posts and users to produce a weighted score that takes their distance into account.

Moreover, in [23], they present an intelligent recruitment platform that uses an ontology-based skill matching algorithm and a complex evaluation to help recruitment sites find the best candidates for specific job openings. Users can post and receive information about job openings by visiting websites. With this graphical ontology model, they calculate the semantic similarity between resumes and employer's requirement for skills and, in order to construct the shortest path's weights summation (semantic similarity) between the appointed pair of skill nodes, they use the classical Dijkstra's algorithm. As part of our strategy, we have created a lengthy list of skills that job seekers may use to enhance their resumes and that employers can utilize when posting job adverts. The matchmaker then takes into account the skills listed on both sides and determines a matching score that will be used to determine the final recommendations.

Furthermore, in [24], they applied a machine learning framework and statistical forecasting models to provide job recommendations. Their approach uses user data along with forecasting and ranking models in order to provide efficient matching between potential candidates and job postings. The authors of [25] presented an approach that aims to provide job recommendations via profile matching methods such as semantic matching, tree-based knowledge matching and query matching. The degree of profile similarity is determined by integrating these methods in accordance with representations of the attributes of both users and jobs. The authors of [26] presented a self-learning recommendation engine that auto-fills missing information in a user's resume by using semantic reasoning. By doing this, it helps to boost a job seeker's chances, having an incomplete resume, of getting more matched job opportunities. Moreover, in [27], they propose a feature selection method using actual data to argue on the significance of the attributes needed for job matching. After the attributes are identified, the proposed system is instructed to use a clustering method to compare the job seekers' profiles to the job postings made by potential employers. Finally, in [28], a system that integrates neural networks and fuzzy logic is presented. The system is trained based on a large set of historical data of unemployed people that were either rejected or approved at several jobs in the past. The process of matching an unemployed person with an offered job is then performed using inference techniques.

3. IMMERSE Architecture

This section provides a high level overview of the architectural design of the IMMERSE system. Figure 1 describes the logical design of the IMMERSE platform, which shows how the different components are organized. The platform is comprised of a number of components, each of which has a specific function to perform. These are the following: (a) the ***Data Management System*** (DMS) component, which is in charge of efficiently storing heterogeneous data, (b) the ***User Interface*** (UI), which serves as the system's primary entry point, (c) the ***Authentication component*** (AUTH), which is used to secure the numerous operations carried out within the IMMERSE system, thereby enhancing data protection, (d) the ***Knowledge Base Service***, (KBS) which primarily accesses the IMMERSE ontology, and (e) a ***Message Bus*** (MSB), which serves as the primary channel for all intercomponent communication.

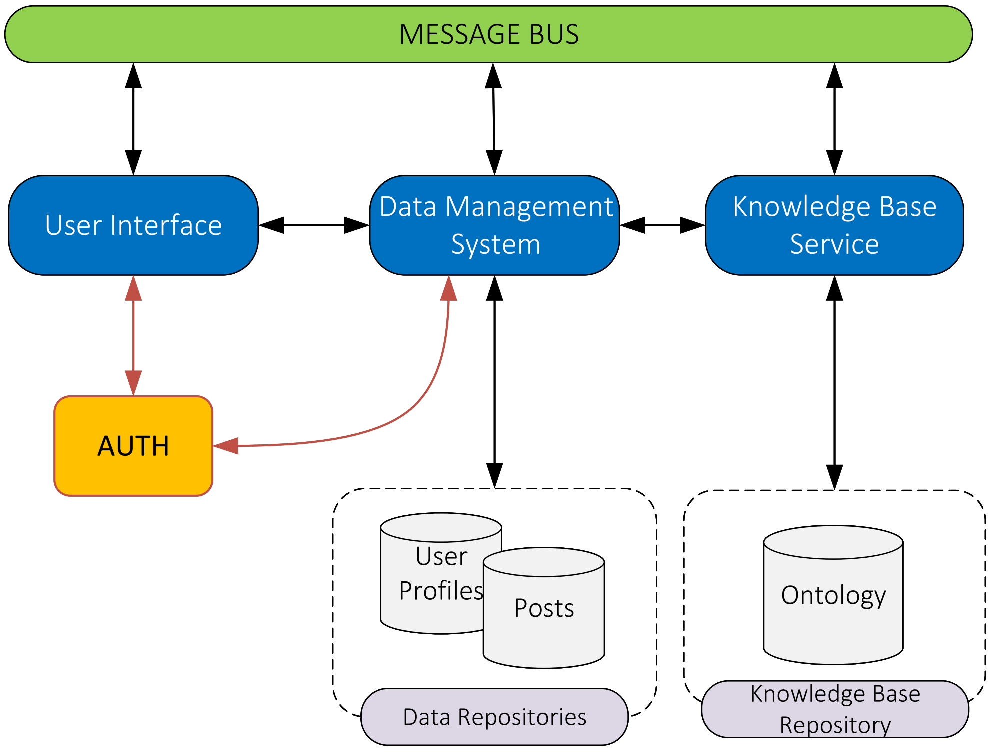


Figure 1. The IMMERSE architecture.

Semantic Framework

The ***Knowledge Base Service*** is a central component of the IMMERSE architecture. It is implemented in Java and acts as the main interface to the IMMERSE ***Knowledge Base*** (KB) also known as ontology, which is described in more detail in Section 4. KBS features subscribe capabilities to the MSB, through which it communicates with DMS for the exchange of information. Its purpose is to update DMS with reasoning results once they become available by its reasoning module.

More specifically, as shown in Figure 2, KBS encapsulates two modules: (a) the ***Knowledge Base Population*** component (KBP), which is responsible for integrating raw data that are collected from DMS, and semantically represent them to the IMMERSE ontology, and (b) the ***Semantic Reasoning*** component (SR) that consists of techniques, rules and algorithms that are performed on top of the IMMERSE domain ontology and aim at deriving new facts and relationships by combining existing knowledge.

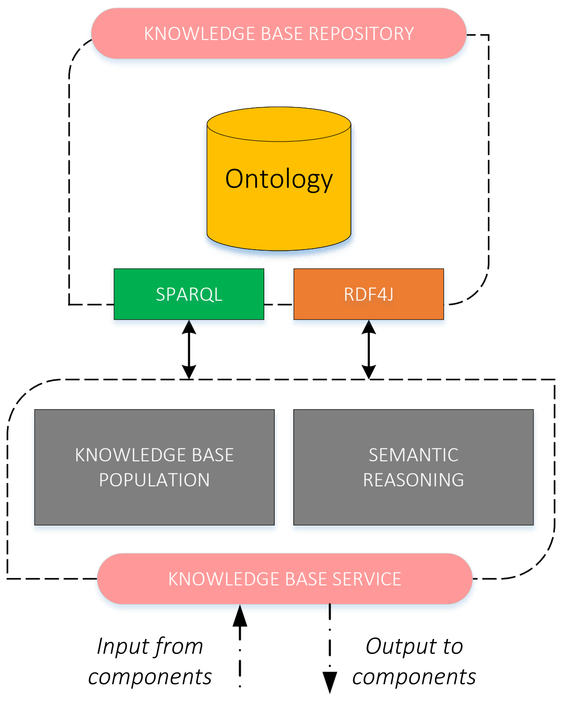


Figure 2. The semantic framework.

4. Ontology

The domain ontology is a lightweight knowledge representation model that is capable of semantically representing:

- *Basic user profile information*, such as location and contact details;
- *Extended user profile information*, including current and previous work experience, spoken languages, educational background, skills and work expectations;
- *Information about service providers*, including their professional profile, contact details as well as information about which services they offer;
- *Information about the actual listings* offered on the platform by its authorized service providers (i.e., job posts);
- *Information about the users' activity* on the platform including their applications and favourite posts.

Moreover, the ontology contains the analysis results that are generated based on the reasoning performed on top of the previous data. Such results contain:

- *Recommendation results* for registered migrants considering their profile and in-app activity as well as the post descriptions and requirements;
- *Similarity relations* between posts of the same category considering their title, description and the populated post attributes.

4.1. Representing User Profiles

The representation of the profile of a migrant user in the ontology is illustrated in Figure 3. More specifically, the ***MigrantProfile*** class refers to information relevant to the migrant's ***resume***. Five classes have been used to represent the migrant's resume (a) ***Language***, (b) ***Education***, (c) ***Work Experience***, (d) ***Skill*** and (e) ***Expectation***, each associated with a set of data properties as shown in the figure.

In addition, each migrant user can have several applications and several saved posts. This information is represented using the object properties ***hasAppliedPost*** and ***hasSavedPost*** respectively that point to an object of type ***Post*** in the ontology. For example, a migrant user can apply to or save posts of type ***Job***.

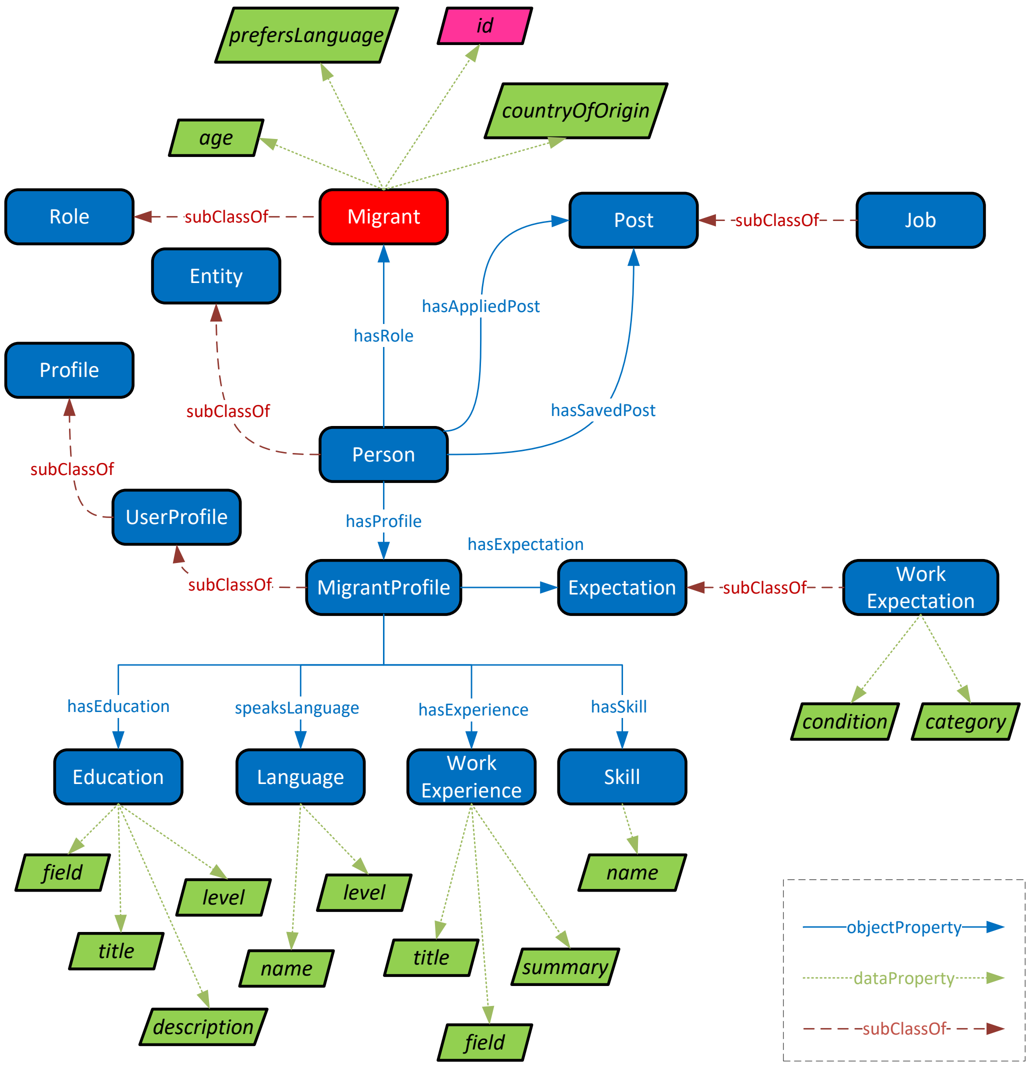


Figure 3. Migrant user representation.

4.2. Representing Job Listings

The class ***Job*** is used to represent a job that is posted in the platform. It is a subclass of the class ***Post*** from which it inherits two data properties, its ***title*** and ***description,*** while it has four other data properties dedicated to the class ***Job*** as depicted in Figure 4. Each job has a ***Category*** where several instances of this type are instantiated in the ontology. Finally, each ***Job*** is linked with other similar jobs through the property ***hasSimilarity*** that points to a class of type ***Similarity***. For each similar job, the ontology contains a property called ***score*** and an object property called ***pointsTo*** that points to a similar job in the domain ontology.

4.3. Representing Job Recommendations

Migrant users can receive recommendations based on the information that they have provided in their profiles. The class ***Job Recommendation*** in Figure 5 is a subclass of the class ***Recommendation***. Recommendations are generated and a score is calculated based on the user's (a) education (property ***education score***), (b) work experience (property ***work exp. score***), (c) spoken languages (property ***language score***), (d) expectations from using the platform, defined by users while creating their profile, such as finding a full time job under a specific category (property ***expectation score***), (e) distance from the job's location (property ***distance score***) and finally (f) a score based on the similarity of the given post to other posts that the user has either applied to or saved for later (property ***similarity score***).

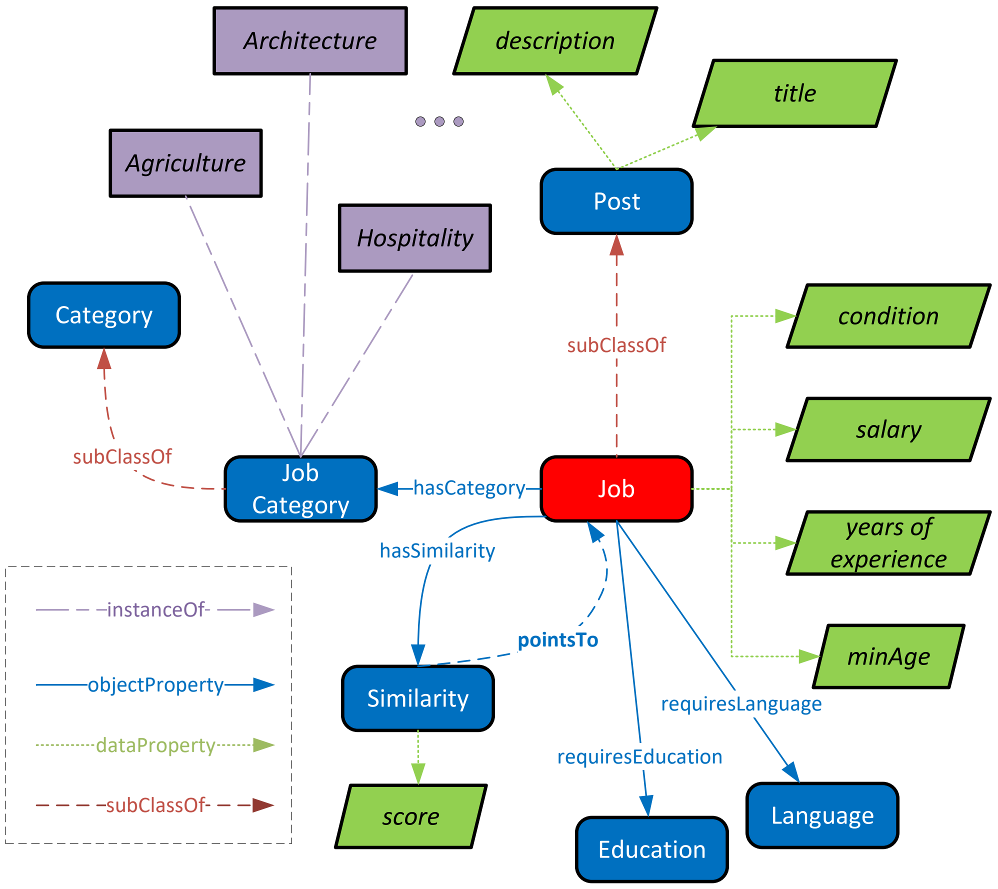


Figure 4. Job representation.

4.4. Ontology Validation

OntoMetrics [29], an online framework that validates ontologies based on established metrics, was used to assess the structure of the IMMERSE ontology. The results of OntoMetrics' analysis are presented in Table 1. Simple metrics, such as the count of classes, axioms, and objects, are included in ***Base Metrics***; these metrics show the number of ontology elements. ***Schema metrics***, on the other hand, are concerned with the ontology's design; metrics in this category indicate the ontology's richness, width, depth, and inheritance.

Starting with the base metrics, the total count of classes and properties indicates that the proposed ontology is a rather lightweight model. In addition, Description Logic (DL) expressivity refers to the Description Logic's variant the ontology belongs to. Since there are many varieties of DLs, there is an informal naming convention, roughly describing the operators allowed [30]. $SI^{(D)}$ expressivity of the IMMERSE ontology indicates a simple ontology, where S means that the ontology contains properties that have a transitive role, I means that the ontology contains inverse properties, and (D) refers to an ontology that uses datatype properties, data values or data types. Regarding schema metrics, the measurements in the table are adopted from [31,32]. More details on each metric are given below:

- ***Attribute richness*** is defined as the average number of attributes per class and can indicate both the quality of ontology design and the amount of information pertaining to instance data. The more attributes that are defined the more knowledge the ontology conveys;

- ***Inheritance richness*** is defined as the average number of subclasses per class and is a good indicator of how well knowledge is grouped into different categories and subcategories in the ontology;
- ***Relationship richness*** refers to the ratio of the number of non-inheritance relationships (i.e., object properties, equivalent classes, disjoint classes) divided by the total number of inheritance (i.e., subclass relations) and non-inheritance relationships defined in the ontology. This metric reflects the diversity of the types of relations in the ontology;
- Finally, ***axiom/class*** ratio, ***class/relation*** ratio, and ***inverse*** relations ratio describe the ratio between axioms–classes, classes–relations, and inverse relations–relations, respectively, and are indications of the ontology's transparency and understandability.

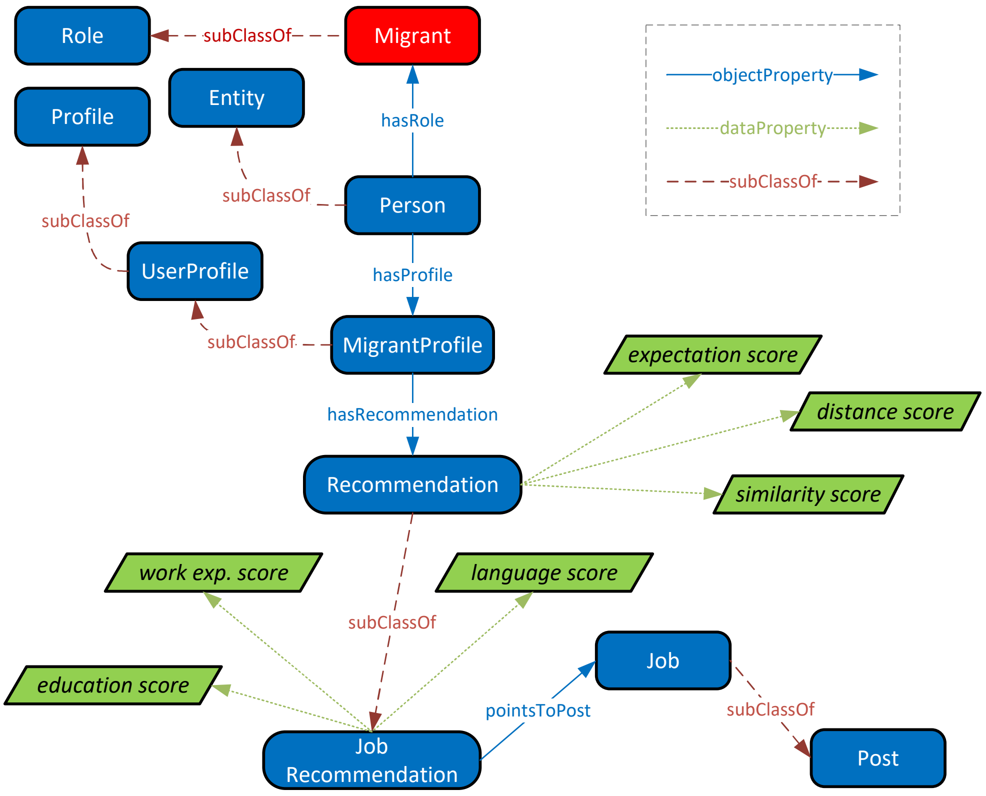


Figure 5. Recommendation representation.

Table 1. Ontology metrics for the IMMERSE ontology are generated by OntoMetrics.

Category	Metric	Value
Basic	Class Count	49
Basic	Object property count	25
Basic	Data property count	84
Basic	Description Logic expressivity	$SI^{(D)}$
Schema	Attribute richness	1.714
Schema	Inheritance richness	0.775
Schema	Relationship richness	0.479
Schema	Axiom/class ratio	9.040
Schema	Inverse relations ratio	0.071
Schema	Class/relation ratio	0.671

5. Semantic Reasoning

Semantic reasoning for IMMERSE is triggered by the Knowledge Base Service whenever new data become available in the system (e.g., when a new user profile is created, a new job is posted or any of the existing profiles and posts are updated). The main purpose of the semantic reasoning framework is to combine, integrate and semantically interpret the knowledge captured in the ontology and eventually extends the semantics of the system using rules and algorithms that are based on the available content, which further updates the knowledge captured in the semantic repository.

The core elements of the reasoning framework are depicted in Figure 6. To host the IMMERSE domain ontology, we have selected the GraphDB graph database [33]. GraphDB is a semantic graph database, compliant with the World Wide Web Consortium (W3C) standards. It is a highly efficient and robust RDF triplestore that provides native OWL 2 RL reasoning services as well as SPARQL-based query interfaces. One important aspect of GraphDB is that it implements the RDF4J framework [34], which is an open-source Java framework, widely used by the semantic community, for working with RDF data, including parsing, storing, inferencing, and querying.

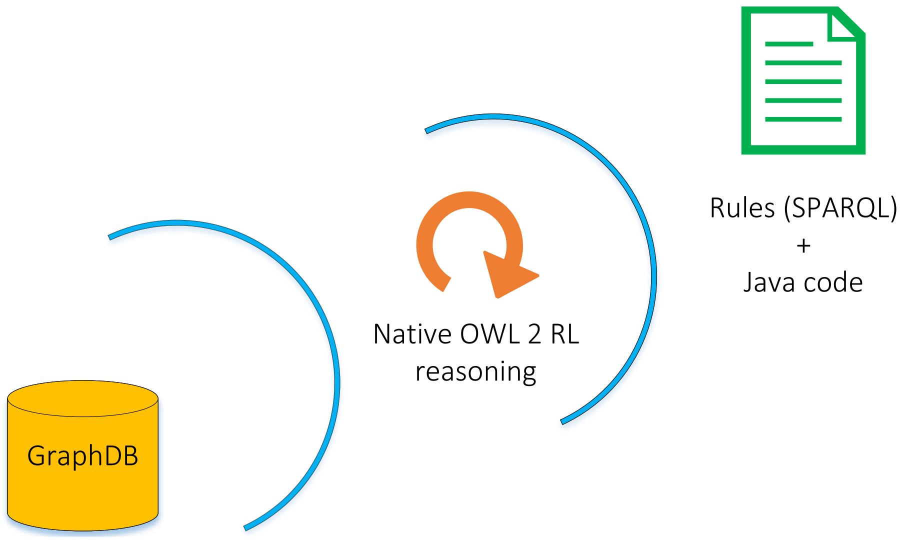


Figure 6. Abstract reasoning architecture.

The OWL 2 RL [35] reasoning, supported by GraphDB, ensures that the properties and characteristics of the OWL 2 language are fully supported. However, OWL 2 reasoning supports limited expressivity and is not able to handle complex domain relations and rules. In addition, it does not allow the dynamic generation of new instances in the knowledge repository. Considering those limitations, we also use a combination of SPARQL queries and Java code (i.e., using the RDF4J framework) to support more advanced and complex relations and enhance the reasoning capabilities of the system. The main idea is to associate each reasoning task with a set of SPARQL queries and algorithms to address specific reasoning requirements. In the following section, we present the requirements that are relevant to the semantic reasoning framework and then the reasoning tasks that were implemented based on the requirements.

Requirements and Competency Questions

A competency question (CQ) is a natural language sentence that expresses a pattern for a type of question people expect an ontology to answer [36]. The answerability of CQs hence becomes a functional requirement of the ontology. This section first provides an overview of the requirements that drove the development process of the proposed reasoning framework and then a list of CQs that the IMMERSE ontology should be able to respond to. Table 2 contains an indicative subset of the reasoning requirements.

Table 2. Requirements relevant to Semantic Reasoning.

Requirement Number	Description
SR_EMPL_01	System should support semantic matchmaking and dynamic discovery of new relationships by automatically analysing the existing content.
SR_EMPL_02	Migrant Users should be able to review matched opportunities, i.e., Jobs matching their CV and in-app interactions.

Based on the list of user requirements above, the ontology is able to respond to several CQs. Table 3 below presents a list of CQs that the IMMERSE Ontology should be able to respond to considering the requirements in Table 2.

Table 3. Competency Questions relevant to Semantic Reasoning.

Code	Description
CQ_01	Which job posts could be recommended to a migrant?
CQ_02	Which job posts match the migrant's language?
CQ_03	Which job posts match the migrant's education?
CQ_04	Which job posts match the migrant's work experience?
CQ_05	Which job posts match the migrant's skills?
CQ_06	Which job posts match the migrant's expectations for finding work?
CQ_07	Which job posts are similar to each other?
CQ_08	Which job posts are similar to posts that the migrant has applied for or saved as a favorite?
CQ_09	Which job posts are closer to the user's location?

The reasoning that is performed is mostly rule-based, meaning that, considering the reasoning requirements, a different set of SPARQL queries is executed along with appropriate Java code. It consists of several individual tasks (see Figure 7), each related to a particular SPARQL rule set. The intermediate results of each task are stored in the IMMERSE domain ontology and then combined to calculate the final recommendations. The role of the KBS is to select which of them should be triggered and execute them. The general goal of the matchmaking process is to recommend posts to users that may be interested in or posts for which the users are appropriate candidates. It is worth mentioning that these tasks can be performed in two cases:

- ***At the user level***, meaning that, given a single user profile (i.e., new or updated), the system will try to find all possible matches against all available posts of a relevant service or,
- ***At the post level***, meaning that, given an individual post (i.e., new or updated) of a service, the system will try to find all possible matches of this post against all relevant profiles.

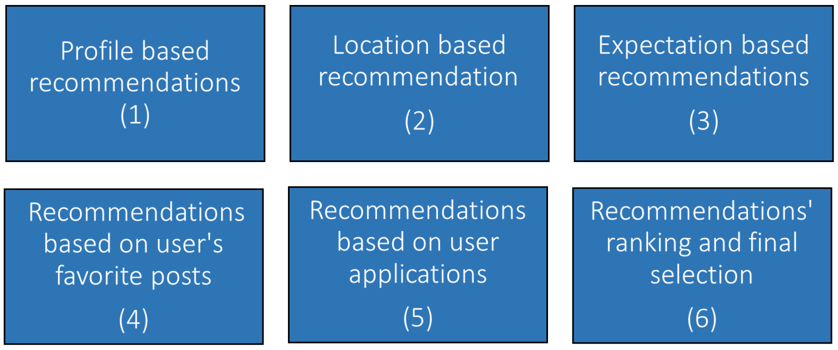

Profile based recommendations (1)
Location based recommendation (2)
Expectation based recommendations (3)
Recommendations based on user's favorite posts (4)
Recommendations based on user applications (5)
Recommendations' ranking and final selection (6)

Figure 7. Individual reasoning tasks.

In both cases, the recommendation results are captured in the ontology and stored under the RDF graph of the particular user. Once they are stored, they are sent to the Data Management Service (DMS) that will store them along with other information relevant to the user. DMS then informs the User Interface (UI) to make the results visible to the user.

6. Simulation Example

Following the methodology proposed in [36], we translated the list of CQs into respective SPARQL queries [37]. In the rest of this section, we provide more details on the individual reasoning tasks and the queries that are performed by the semantic reasoning component. We are going to demonstrate them through a simulation example. The example assumes that a new user has registered and populated the CV, thus the rules are performed at the ***user level***. In case a new post is introduced in the system, the component automatically adjusts the same rules, and calculations are performed at the ***post level***.

6.1. Initialization Step

The step prior to any other execution of the semantic reasoning component is the initialization of the triples that will be used throughout the calculations. The SPARQL query in Figure 8 populates the appropriate properties and creates a link between the user and the posts offered in the user's country as well as sets all scores to zero.

Once the triples for each potential recommendation are initialized, the first actual reasoning task that is executed concerns the recommendations that the system calculates considering only the profile of the migrant user. This reasoning task is executed in four steps:

- ***Step 1*** considers the languages spoken by the user;
- ***Step 2*** considers the user's education;
- ***Step 3*** considers the user's past work experience and;
- ***Step 4*** considers the user's skills;

6.2. Step 1: Language Recommendations

Through the UI, migrant users can define the languages they speak along with their level, and service providers can define the language requirements of the posts they are creating. This information is used by the semantic reasoning component to find which posts match the profile of the migrant considering the language. Even if the user does not provide any language, the system makes use of the user's preferred language that is provided upon registration. The following SPARQL query in Figure 9 considers only posts that could be potentially recommended to a particular user as those have been calculated during the initialization step. It retrieves all possible combinations of languages spoken by a user and languages required by a post.

For example, assuming that a migrant speaks English and Arabic, and a post requires English and French (see Table 4), then the previous query would return the following combinations:

The next step is to check if there is a perfect match, which means that the languages that are spoken by the user match all the required languages. In this case, the system updates the ***langScore*** property by assigning the calculated score; otherwise, it gives a default equal to -10.0 (i.e., meaning that the user is not a suitable candidate for this post based on the language requirements).

6.3. Step 2: Education Recommendations

Through the UI, migrant users can define their education (i.e., the education field and level of education) and service providers can define the education requirements of the posts they are creating (if any). To find which posts match the migrant's education, the system queries the ontology to obtain all possible combinations of the migrant's education and the education requirement of all posts using the SPARQL query depicted in Figure 10. The rule considers only posts that could be potentially recommended to a particular user as those have been calculated during the initialization step.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
INSERT {
    GRAPH ?g {
        ?s miict:hasRecommendation ?rec_job .
        ?rec_job rdf:type miict:JobRecommendation ;
            miict:pointsToPost ?job ;
            miict:langScore 0.0 ;
            miict:eduScore 0.0 ;
            miict:workScore 0.0 ;
            miict:expScore 0.0 ;
            miict:simScore 0.0 ;
            miict:skillScore 0.0 ;
            miict:distScore 0.0 .
    }
} WHERE {
    GRAPH ?g {
        ?s miict:hasRole miict:Migrant ;
            miict:id ?id ;
            miict:hasLocation ?loc .
            ?loc miict:country ?country .
            ?job rdf:type miict:JobPost ;
                miict:id ?job_id ;
                miict:hasLocation ?job_loc .
            ?job_loc miict:country ?job_country .
            BIND(IRI(CONCAT("miict:", CONCAT(STR(?id),STR(?job_id))))
                AS ?rec_job)
            FILTER(?id = STR("60a6d20d6ec17bc2c599efb9")
                && ?country = ?job_country)
        }
}
```

Figure 8. CQ_01: Which job posts could be recommended to a migrant?

Table 4. Language combinations.

Combination	Spoken Language	Required Language	Score
1	English (Beginner)	English (Beginner)	1.0
2	English (Beginner)	French (Beginner)	0.0
3	Arabic (Advanced)	English (Beginner)	0.0
4	Arabic (Advanced)	French (Beginner)	0.0

For example, assuming that a user has a Master's Degree (i.e., level of education) in Information Technology (i.e., education field), and a post requires a Bachelor's Degree in Information Technology (see Table 5), then the previous query would return the following combination:

The next step is to check if there is a perfect match meaning that the education of the user covers the education requirements of the post. In this case, the system updates the ***eduScore*** property by assigning the calculated score; otherwise, it gives a default value equal to −10.0 (i.e., meaning that the user is not a suitable candidate for this post based on the education requirements).

```
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   SELECT DISTINCT ?r ?name ?level ?rname ?rlevel WHERE {
       ?s miict:hasRole miict:Migrant ;
          miict:id ?id ;
          miict:speaksLanguage ?l .
       ?l miict:languageName ?name ;
          miict:languageLevel ?level .
       ?s miict:hasRecommendation ?r .
       ?r rdf:type miict:JobRecommendation ;
          miict:pointsToPost ?j .
       ?j rdf:type miict:JobPost ;
          miict:id ?job_id ;
          miict:requiresLanguage ?rl .
       ?rl miict:languageName ?rname ;
          miict:languageLevel ?rlevel .
       FILTER(?id = STR("60a6d20d6ec17bc2c599efb9"))
}
```

Figure 9. CQ_02: Which job posts match the migrant's language?

```
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   SELECT DISTINCT ?r ?field ?level ?rfield ?rlevel WHERE {
       ?s miict:hasRole miict:Migrant ;
          miict:id ?id ;
          miict:hasEducation ?e .
       ?e miict:educationField ?field ;
          miict:educationDegreeLevel ?level .
       ?s miict:hasRecommendation ?r .
       ?r rdf:type miict:JobRecommendation ;
          miict:pointsToPost ?j .
       ?j rdf:type miict:JobPost ;
          miict:id ?job_id ;
          miict:requiresEducation ?re .
       ?re miict:educationField ?rfield ;
          miict:educationDegreeLevel ?rlevel .
       FILTER(?id = STR("60a6d20d6ec17bc2c599efb9"))
}
```

Figure 10. CQ_03: Which job posts match the migrant's education?

Table 5. Education combinations.

Combination	Education	Required Education	Score
1	IT (Master)	IT (Bachelor)	1.0

6.4. Step 3: Work Experience Recommendations

Similarly, through the UI, migrant users can define their past work experience (i.e., the field of the jobs they have worked) and service providers can define the job category of the posts they are creating. To find which posts match the migrant's past experience, the system queries the ontology to obtain all possible combinations between the job field of the migrant's past experience and the job field of all posts using the SPARQL query depicted in Figure 11. The rule considers only posts that could be potentially recommended to the particular user as those have been calculated during the initialization step.

```
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    SELECT DISTINCT ?r ?field ?ca_name WHERE {
        ?s miict:hasRole miict:Migrant ;
           miict:id ?id ;
           miict:hasWorkExp ?exp .
        ?exp miict:expField ?field .
        ?s miict:hasRecommendation ?r .
        ?r rdf:type miict:JobRecommendation ;
           miict:pointsToPost ?j .
        ?j rdf:type miict:JobPost ;
           miict:id ?job_id ;
           miict:hasCategory ?ca .
        ?ca miict:categoryName ?ca_name .
        FILTER(?id = STR("60a6d20d6ec17bc2c599efb9"))
}
```

Figure 11. CQ_04: Which job posts match the migrant's work experience?

For example, assuming that a user has worked in the field of Information Technology and Education and Training, and a post is offered under the category Education and Training (see Table 6), then the previous query would return the following combinations:

Table 6. Work Experience Combinations.

Combination	Work Exp.	Required Work Exp.	Score
1	IT	Education & Training	0.0
2	Education & Training	Education & Training	1.0

The last column is the score that is calculated for each combination. More specifically, if any of the past experience matches the job category of the post, then the given score is 1.0. In this case, the system updates the ***workScore*** property by assigning the calculated score. In the case of the past work experience, we do not assign a value of −10.0 if the user has no past experience because this post could be recommended assuming that the user would cover the language and education requirements of the post.

6.5. Step 4: Skills Recommendations

Finally, through the UI, migrant users can define their skills and service providers can define the skill requirements of the posts they are creating. Both sides can select skills from a predefined dropdown list. To find which posts match the migrant's skills, the system queries the ontology to obtain the list of the skills defined by the user along with the list of skill requirements defined for each post using the SPARQL query depicted in Figure 12. The rule considers only posts that could be potentially recommended to the particular user as those have been calculated during the initialization step.

For example, assuming that a user has the skills MS Office and Meeting Deadlines, and a couple of posts require a particular set of skills (see Table 7), then the previous query would return the following combinations:

The last column is the score that is calculated for each combination. More specifically, the score is calculated based on the following equation:

$$score = \frac{\#\ of\ matched\ skills}{\#\ of\ total\ required\ skills} \tag{1}$$

```
PREFIX : <http://www.semanticweb.org/certh/miict>
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    SELECT ?r ?list ?req_list WHERE {
        ?s miict:hasRole miict:Migrant ;
            miict:id ?id ;
            miict:hasRecommendation ?r ;
            miict:hasSkills ?skills .
        ?skills miict:skillList ?list .

        ?r rdf:type miict:JobRecommendation ;
            miict:pointsToPost ?p .
        ?p rdf:type miict:JobPost ;
            miict:id ?job_id ;
            miict:expRequirements ?req .
        ?req miict:skillList ?req_list .
        FILTER (?id = STR("60a6d20d6ec17bc2c599efb9")
        && ?list != "undefined" && ?req_list != "undefined")
}
```

Figure 12. CQ_05: Which job posts match the migrant's skills?

Table 7. Skills' combinations.

Combination	Skills	Required Skills	Score
1	MS Office, Meeting Deadlines	MS Office, Problem Solving	0.5
2	MS Office, Meeting Deadlines	Java	0.0

6.6. Expectations' Recommendations

The panel of migrant users enables them to define the expectations they have from the IMMERSE platform such as finding work in a particular field where they can also define the work condition they prefer (e.g., part-time). To find which posts match the migrant's expectations for work, the system queries the ontology to obtain all possible combinations between the work expectations of the migrant and the job field of all posts using the SPARQL query depicted in Figure 13. The rule considers only posts that could be potentially recommended to the particular country as those have been calculated during the initialization step.

For example, assuming that a user wants to work in the field of Hospitality and Tourism (i.e., full-time) and there are two posts in the field of Hospitality and Tourism, one offering part-time and the other full-time conditions (see Table 8), then the previous query would return the following combinations:

The last column is the score that is calculated for each combination. More specifically, if a given combination matches only the job field but not the work condition, the score is 0.5, but if both job field and work condition match, the given score is 1.0. The level of the language spoken by the migrant needs to be at least equal to or greater than the level of the required language to obtain a score of 1.0.

```
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?r ?cat ?c ?name ?cond WHERE {
    ?s miict:hasRole miict:Migrant ;
        miict:id ?id ;
        miict:hasExpectation ?e .
    ?e rdf:type miict:WorkExpectation ;
        miict:exp_condition ?c ;
        miict:exp_category ?cat .
    ?s miict:hasRecommendation ?r .
    ?r rdf:type miict:JobRecommendation ;
        miict:pointsToPost ?j .
    ?j rdf:type miict:JobPost ;
        miict:id ?job_id ;
        miict:hasCategory ?ca ;
        miict:workCondition ?cond .
    ?ca miict:categoryName ?name .
    FILTER(?id = STR("60a6d20d6ec17bc2c599efb9"))
}
```

Figure 13. CQ_06: Which job posts match the migrant's expectations for finding work?

Table 8. Expectations' Combinations.

Combination	Work Expectation	Offered Jobs	Score
1	Hospitality & Tourism/full-time	Hospitality & Tourism/full-time	0.5
2	Hospitality & Tourism/full-time	Hospitality & Tourism/part-time	0.0

6.7. Similarities' Recommendations

Considering the interactions of the migrant within the IMMERSE platform, the system needs to be able to also recommend similar posts to those that the user has either showed interest in (i.e., saved as a favorite posts) or applied. By similar posts, we mean posts that have similar requirements (e.g., similar language or past experience requirements). Thus, it is important that the system can identify those posts which are similar. This section provides information on how the system calculates the similarities between different posts. To find which posts are similar based on their requirements, the system first queries the ontology to get all requirements of the posts using the SPARQL query depicted in Figure 14. In our example, those would be the job category, languages, education, country, and work condition. The query considers all job posts that are available on the platform.

The values of the ***category*** and the ***country*** are used by the system to compare only jobs of the same category that are offered in the same country. Then, each post is compared against all other posts and a score is calculated. For example, assuming that we have the following two job posts having the values shown in Table 9, we can see that 3 out 5 properties have the same values; thus, the final score of their similarity is 0.6.

Eventually, every time a new post is added to the platform, the reasoning component first calculates which posts are similar to it. It then starts re-calculating recommendations based on favourite posts and applications for each user living in the same country. The following SPARQL rule, depicted in Figure 15, queries the ontology to find all the posts that the migrant user has either saved or applied and also finds which posts are similar to them. For each similar post, it also queries the final similarity score and uses it to update the property ***simScore*** that holds the score of the recommendation object.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
SELECT DISTINCT ?category ?language_name ?l_level ?ed_field ?ed_level
    ?condition {
    ?s rdf:type miict:JobPost;
        miict:id ?id ;
        miict:jobName ?name ;
        miict:jobDescription ?description ;
        miict:hasCategory ?c ;
        miict:hasLocation ?loc .
    ?c miict:categoryName ?category .
    ?loc miict:country ?country .
    ?s miict:requiresEducation ?education .
    ?education miict:educationField ?ed_field .
    ?education miict:educationDegreeLevel ?ed_level .
    OPTIONAL {
        ?s miict:requiresLanguage ?language .
        ?language miict:languageName ?language_name .
        ?language miict:languageLevel ?l_level .
    }
    ?s miict:workCondition ?condition .
}
```

Figure 14. CQ_07: Which job posts are similar to each other?

Table 9. Post similarity combinations.

Property	Job #1	Job #2	Score
Category	Human Services	Human Services	-
Country	Spain	Spain	-
Condition	Full-Time	Part-Time	0.0
Education Level	High School	High School	1.0
Education Field	No Requirement	No Requirement	1.0
Language #1	English	English	1.0
Language #2	French	-	0.0
		Total Score	0.6 (=3/5)

For example, assuming that a user has one saved post (Post #1) and applied to another post (Post #2) where Post #1 is similar to Post #3 with a score of 0.5 and with Post #4 with a score of 0.3. In addition, consider that Post #2 is also similar to Post #3 with a score of 0.4. Then, based on the previous query, as shown in Table 10, Post #3 will have a recommendation score (i.e., based on its similarity with posts that the user has interacted with) equal to 0.45 (=0.5 + 0.4/2), which is the average value of the two similarity scores, and Post #4 will have a recommendation score equal to 0.3.

6.8. Distance Recommendations

Each post that is created in the IMMERSE platform and each user that registers has a location. Even if no specific address is provided by the migrant user, the system knows at least the country where the user is located since this information is provided upon registration from all users. Based on those two locations, the system can calculate the distance between users and posts, thus giving more weight to posts that are closer to the user. Table 11 summarizes the distance ranges that are considered by the system and the scores that are assigned to each case.

```
PREFIX miict: <http://www.semanticweb.org/certh/miict#>
DELETE {?r miict:simScore ?old_score}
INSERT {
   GRAPH miict:60a6d20d6ec17bc2c599efb9
   {
      ?r miict:simScore ?avg
   }
} WHERE {
   ?s miict:hasRole miict:Migrant ;
      miict:id ?id .
   ?s miict:hasRecommendation ?r .
   ?r rdf:type miict:JobRecommendation ;
      miict:pointsToPost ?post .
   ?r miict:simScore ?old_score .
   {
      SELECT ?post (AVG(?score) as ?avg) WHERE {
         ?s miict:hasRole miict:Migrant ;
            miict:id ?id ;
            miict:hasAppliedJobs ?applied ;
            miict:hasSavedJobs ?saved ;
            miict:hasRecommendation ?r .
         ?r rdf:type miict:JobRecommendation ;
            miict:pointsToPost ?p ;
            miict:simScore ?old_score .
         ?p rdf:type miict:JobPost ;
            miict:id ?job_id .
         {
            BIND (?saved AS ?posts)
         }
         UNION
         {
            BIND(?applied AS ?posts)
         }
         ?posts miict:hasSimilarity ?similarity .
         ?similarity miict:pointsToPost ?post .
         ?similarity miict:finalScore ?score .
         FILTER(?id = STR("60a6d20d6ec17bc2c599efb9"))
      } GROUP BY ?post
   }
}
```

Figure 15. CQ_08: Which job posts are similar to posts that the migrant has applied for or saved as a favorite?

Table 10. Similarity combinations.

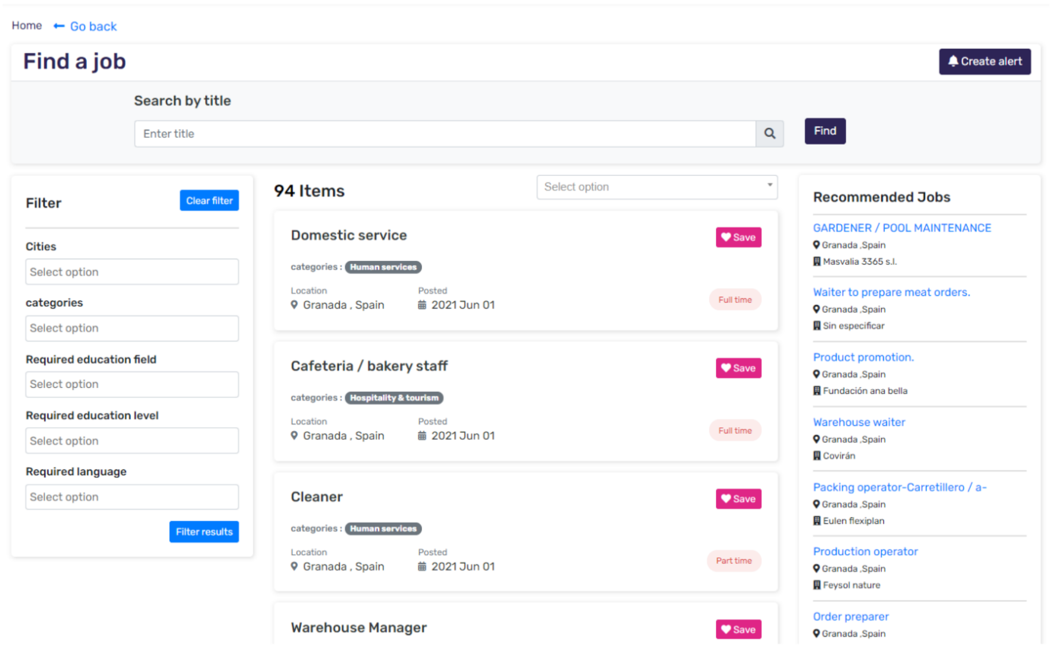
Saved or Applied Post	Similar Posts	Similarity Score
Post #1	Post #3	0.5
Post #1	Post #4	0.3
Post #2	Post #3	0.4

Table 11. Distance ranges.

Distance Range (in km)	Distance Score
[0, 6)	1.0
[6, 15)	0.7
[15, 35)	0.5
[35, 100)	0.2
[100,)	0.0

6.9. Matchmaking Results in the User Interface

Once the scores of the individual tasks are calculated, the system calculates the final recommendation scores by adding intermediate scores and sorts them in descending order. The top 10 posts are then sent back to the DMS through a post request. The body of the request is a JSON array having the IDs of the posts that need to be displayed to the UI. Those IDs are stored in the database of the DMS and then the UI retrieves additional information about those posts and provides it to the user while the user navigates through the respective service. Figure 16 depicts how the results of the reasoning component are displayed in the UI (i.e., in the Recommended Jobs sidebar).



Domestic service
categories : Human services
Location Granada , Spain
Posted 2021 Jun 01

Cafeteria / bakery staff
categories : Hospitality & tourism
Location Granada , Spain
Posted 2021 Jun 01

Cleaner
categories : Human services
Location Granada , Spain
Posted 2021 Jun 01

Warehouse Manager

Recommended Jobs

GARDENER / POOL MAINTENANCE
Granada ,Spain
Masvalia 3365 s.l.

Waiter to prepare meat orders.
Granada ,Spain
Sin especificar

Product promotion.
Granada ,Spain
Fundación ana bella

Warehouse waiter
Granada ,Spain
Covirán

Packing operator-Carretillero / a-
Granada ,Spain
Eulen flexiplan

Production operator
Granada ,Spain
Feysol nature

Order preparer
Granada ,Spain

Figure 16. Job recommendations in the UI.

7. Experimental Results

In this section, the proposed recommendation framework is evaluated in a real-world dataset, and the results of the evaluation are presented.

7.1. Dataset Description

To evaluate the responsiveness and scalability of the proposed reasoning framework, a real-world dataset was considered. The dataset was created during the piloting phase of the IMMERSE platform and consists of 100 real job posts and 30 user profiles where users have included in their profile information about their languages, education, work experience, skills, etc.

7.2. Evaluating the Recommender System

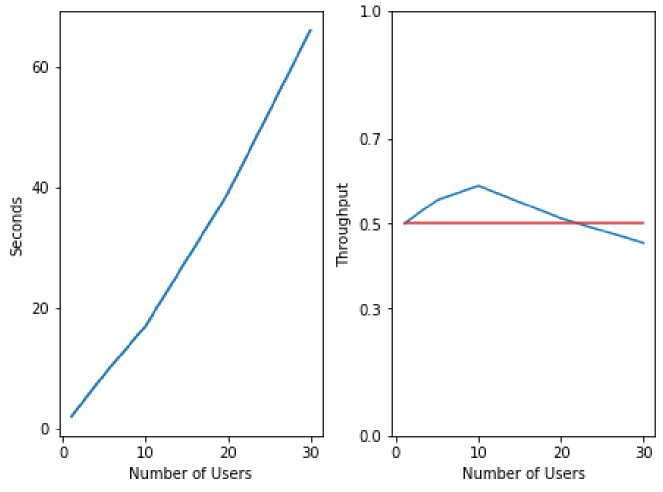
To measure the response time and see how well our system scales while increasing its processing demand, we decided to split the evaluation process into four phases. In each phase, we would gradually increment the number of available posts and then, for every phase, we would also increment the number of concurrent users for which the system would have to calculate recommendations. To measure the scalability of the system, we used a metric called *throughput*, which is defined as the number of recommendations per second.

Throughput is calculated using the following equation:

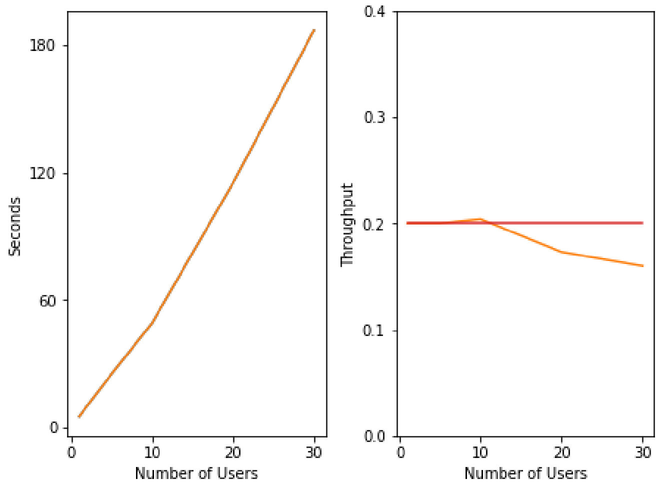
$$X = N/R, \tag{2}$$

where N is the number of concurrent users, and R is the average time the system needs to completed its calculations.

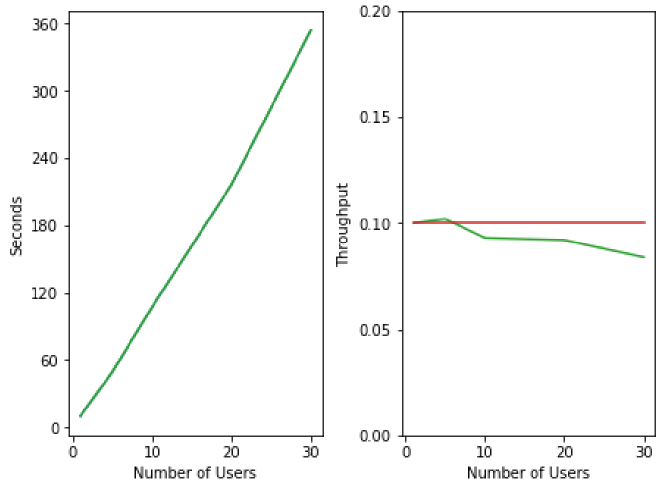
A good comparison against which we could compare our system would be a *linearly scalable* version of our system, meaning a system that continues to do exactly the same amount of work per second no matter how many users are using it. This does not mean that the system's response time will remain constant. In fact, it will increase but in a perfectly predictable manner. However, its throughput will remain constant. Linearly scalable applications are perfectly scalable in that their performance degrades at a constant rate directly proportional to their demands. The results of our evaluation are depicted in Figure 17.



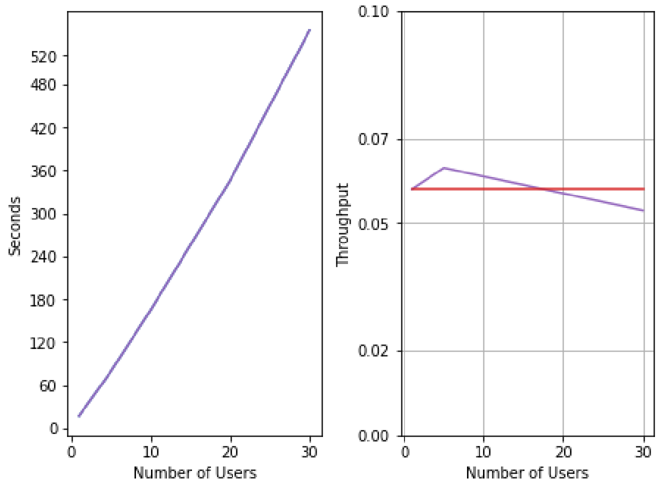
(20 posts)



(50 posts)



(80 posts)



(100 posts)

Figure 17. Reasoning response time and scalability.

8. Conclusions

In this paper, we have discussed and demonstrated an ontology-based reasoning framework acting as a matching tool that enables the contexts of individual migrants and refugees, including their expectations, languages, educational background, previous job experience and skills, to be captured in the ontology and facilitate their matching with the job opportunities available in their host country. We have presented the main components of the IMMERSE reasoning framework and how this has been incorporated in the IMMERSE main architecture. In addition, we examined scalability and reported on computational performance as a function of the number of jobs and job seekers available in the knowledge base repository. Our results show that the system scales linearly and can be optimized for large scale implementation. Our current implementation focuses on finding appropriate jobs based on academic qualifications, languages, skills and work experience. For future work, the framework could be extended to consider several other constraints that exist when seeking jobs and these are often subjective i.e., highly dependent on the job seeker in question. Such factors are location flexibility that includes both the option to work remotely or being open to relocation, job seeker's age, type of companies the job seeker has worked with in the past (small, medium, or large size), industry, etc.

Author Contributions: Conceptualization, D.N., A.K. and G.M.; methodology, D.N., A.K. and G.M.; software, D.N., P.M. and G.M.; validation, D.N., S.V. and G.M.; formal analysis, D.N.; investigation, D.N. and G.M.; writing—original draft preparation, D.N., P.M. and G.M.; writing—review and editing, S.V. and I.K.; visualization, D.N.; supervision, S.V. and I.K.; project administration, S.V. and I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work has received funding by the EC under contracts H2020-822380 MIICT and H2020-870939 SO-CLOSE.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Patricia Bueso and Jose Gonzalez-Riera from the Euro-Arab Foundation for Higher Studies (FUNDEA) who organized the pilots where the data used for the evaluation of the proposed framework were collected.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IMMERSE	Integration of Migrants MatchER Service
MIICT	ICT Enabled Public Services for Migration
CV	Curriculum Vitae
RDF	Resource Description Framework
OWL	Web Ontology Language
FOAF	Friend of a Friend
DOAC	Description of a Career
SPARQL	Simple Protocol and RDF Query Language
DMS	Data Management System
UI	User Interface
AUTH	Authentication
KBS	Knowledge Base Service
MSB	Message Bus
KB	Knowledge Base
KBP	Knowledge Base Population
SR	Semantic Reasoning
DL	Description Logic
W3C	World Wide Web Consortium
CQ	Competency Question

References

1. Zimmermann, K.F. Refugee and migrant labor market integration: Europe in need of a new policy agenda. In *The Integration of Migrants and Refugees. An EUI Forum on Migration, Citizenship and Demography*; European University Institute, Robert Schuman Centre for Advanced Studies: Florence, Italy, 2017.
2. AbuJarour, S.; Wiesche, M.; Andrade, A.D.; Fedorowicz, J.; Krasnova, H.; Olbrich, S.; Tan, C.W.; Urquhart, C.; Venkatesh, V. ICT-enabled refugee integration: A research agenda. *Commun. AIS* **2019**, *44*, 874–891. [CrossRef]

3. AbuJarour, S.; Krasnova, H.; Hoffmeier, F. ICT as an enabler: Understanding the role of online communication in the social inclusion of Syrian refugees in Germany. In Proceedings of the Twenty-Sixth European Conference on Information Systems (ECIS 2018), Portsmouth, UK, 23–28 June 2018.
4. Vernon, A.; Deriche, K.; Eisenhauer, S. *Connecting Refugees—HOW INTERnet and Mobile Connectivity Can Improve Refugee Well-Being and Transform Humanitarian Action*; UNHCR: Geneva, Switzerland, 2016.
5. Andrade, A.D.; Doolin, B. Information and communication technology and the social inclusion of refugees. *Mis Q.* **2016**, *40*, 405–416. [CrossRef]
6. ICT Enabled Public Services for Migration. Available online: https://www.miict.eu/ (accessed on 16 September 2022).
7. Ntioudis, D.; Karakostas, A.; Vrochidis, S.; Kompatsiaris, I. IMMERSE: A Matching Platform Improving Migrant Integration with Semantic Technologies. In *Information and Communications Technology in Support of Migration*; Springer: Cham, Switzerland, 2022; pp. 213–228.
8. Ntioudis, D.; Kamateri, E.; Meditskos, G.; Karakostas, A.; Huber, F.; Bratska, R.; Vrochidis, S.; Akhgar, B.; Kompatsiaris, I. Immerse: A personalized system addressing the challenges of migrant integration. In Proceedings of the 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), London, UK, 6–10 July 2020; pp. 1–6.
9. Decker, S.; Melnik, S.; Van Harmelen, F.; Fensel, D.; Klein, M.; Broekstra, J.E.; Erdmann, M.; Horrocks, I. The Semantic Web: The roles of XML and RDF. *IEEE Internet Comput.* **2000**, *15*, 63–74. [CrossRef]
10. Ma, L.; Yang, Y.; Qiu, Z.; Xie, G.; Pan, Y.; Liu, S. Towards a complete OWL ontology benchmark. In *European Semantic Web Conference*; Springer: Berlin/Heidelberg, Germany, 2006, pp. 125–139.
11. World Wide Web Consortium (W3C). Available online: https://www.w3.org/ (accessed on 28 September 2022).
12. Brickley, D.; Miller, L. FOAF Vocabulary Specification 0.91. 2007. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.476.8247&rep=rep1&type=pdf (accessed on 28 July 2022).
13. Core Vocabularies. Available online: https://ec.europa.eu/isa2/solutions/core-vocabularies_en/ (accessed on 25 July 2022).
14. Ahmed, N.; Khan, S.; Latif, K. Job description ontology. In Proceedings of the 2016 International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 19–21 December 2016; pp. 217–222.
15. Parada, R. Doac Vocabulary Specification. 2006. Available online: https://www.edatasoft.com/en/doac/01/ (accessed on 28 July 2022).
16. Fazel-Zarandi, M.; Fox, M.S. An ontology for skill and competency management. In Proceedings of the 7th International Conference on Formal Ontology in Information Systems, Graz, Austria, 24–27 July 2012; pp. 89–102.
17. Bojārs, U.; Breslin, J.G. ResumeRDF: Expressing skill information on the Semantic Web. In Proceedings of the 1 st International ExpertFinder Workshop, Berlin, Germany, 16 January 2007.
18. Gómez-Pérez, A.; Ramírez, J.; Villazón-Terrazas, B. An ontology for modelling human resources management based on standards. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 534–541.
19. Harris, S.; Seaborne, A.; Prud'hommeaux, E. SPARQL 1.1 query language. *W3C Recomm.* **2013**, *21*, 778.
20. Fazel-Zarandi, M.; Fox, M.S. Semantic matchmaking for job recruitment: An ontology-based hybrid approach. In Proceedings of the 8th International Semantic Web Conference, Chantilly, VA, USA, 25–29 October 2009; Volume 525, p. 2009.
21. Malinowski, J.; Keim, T.; Wendt, O.; Weitzel, T. Matching people and jobs: A bilateral recommendation approach. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), Kauai, HI, USA, 4–7 January 2006; Volume 6, p. 137c.
22. Lee, D.H.; Brusilovsky, P. Fighting information overflow with personalized comprehensive information access: A proactive job recommender. In Proceedings of the Third International Conference on Autonomic and Autonomous Systems (ICAS'07), Athens, Greece, 19–25 June 2007; p. 21.
23. Lv, H.; Zhu, B. Skill ontology-based semantic model and its matching algorithm. In Proceedings of the 2006 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, Hangzhou, China, 17–19 November 2006; pp. 1–4.
24. Kenthapadi, K.; Le, B.; Venkataraman, G. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 346–347.
25. Musale, D.V.; Nagpure, M.K.; Patil, K.S.; Sayyed, R.F. Job recommendation system using profile matching and web-crawling. *Int. J.* **2016**, *1*. Available online: http://www.ijasret.com/VolumeArticles/FullTextPDF/24_IJASRET7747.pdf (accessed on 28 July 2022).
26. Koh, M.F.; Chew, Y.C. Intelligent job matching with self-learning recommendation engine. *Procedia Manuf.* **2015**, *3*, 1959–1965. [CrossRef]
27. Rodriguez, L.G.; Chavez, E.P. Feature selection for job matching application using profile matching model. In Proceedings of the 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), Singapore, 23–25 February 2019; pp. 263–266.
28. Drigas, A.; Kouremenos, S.; Vrettos, S.; Vrettaros, J.; Kouremenos, D. An expert system for job matching of the unemployed. *Expert Syst. Appl.* **2004**, *26*, 217–224. [CrossRef]
29. OntoMetrics. Available online: https://ontometrics.informatik.uni-rostock.de/ontologymetrics/ (accesses on 19 September 2022).
30. Description Logic. Available online: https://en.wikipedia.org/wiki/Description_logic (accessed on 19 September 2022).

31. Falco, R.; Gangemi, A.; Peroni, S.; Shotton, D.; Vitali, F. Modelling OWL ontologies with Graffoo. In *European Semantic Web Conference*; Springer: Cham, Switzerland, 2014; pp. 320–325.
32. Tartir, S.; Arpinar, I.B.; Sheth, A.P. Ontological evaluation and validation. In *Theory and Applications of Ontology: Computer Applications*; Springer: Dordrecht, The Netherlands, 2010; pp. 115–130.
33. GraphDB. Available online: https://graphdb.ontotext.com/documentation/10.0/about-graphdb.html (accessed on 19 September 2022).
34. Eclipse RDF4J framework. Available online: https://rdf4j.org/ (accessed on 19 September 2022).
35. OWL 2 Web Ontology Language Profiles (Second Edition). Available online: https://www.w3.org/TR/owl2-profiles/#OWL_2_RL (accessed on 19 September 2022).
36. Uschold, M.; Gruninger, M. Ontologies: Principles, methods and applications. *Knowl. Eng. Rev.* **1996**, *11*, 93–136. [CrossRef]
37. Zemmouchi-Ghomari, L.; Ghomari, A.R. Translating natural language competency questions into SPARQL queries: A case study. In Proceedings of the First International Conference on Building and Exploring Web Based Environments, Seville, Spain, 27 January–1 February 2013; pp. 81–86.