PRAGUE UNIVERSITY OF ECONOMICS AND BUSINESS

Acta Informatica Pragensia
2023, Volume 12, Issue 2, pp. 200–224
https://doi.org/10.18267/j.aip.204

Article Open Access

AnnoJOB: Semantic Annotation-Based System for Job Recommendation

Assia Brek , Zizette Boufaida

LIRE Laboratory, Software and Information Systems Technologies Department, Faculty of Information and Communication Technology, Constantine 2 University – Abdelhamid Mehri, Constantine, Algeria

Corresponding author: Assia Brek (assia.brek@univ-constantine2.dz)

Abstract

With the vast success of e-recruitment, online job offers have increased. Therefore, there is a number of job portals and recommendation systems trying to help users filter this massive amount of offers when searching for the right job. Until today, most of these systems' searching techniques are confined to using keywords such as job titles or skills, which also returns many results. This paper proposes a job recommender system that exploits the candidate's resume to select the appropriate job. Our system, AnnoJob, adopts a semantic annotation approach to: (1) intelligently extract contextual entities from resumes/offers, and (2) semantically structure the extracted entities in RDF triples using domain ontology, providing a unified presentation of the content of the documents. Furthermore, to select the suitable offer, we propose a novel semantic matching technique that computes the similarity between the resume/offers based on identifying the semantic similarity and relatedness between the RDF triples using the domain ontology and Wikidata, which enhance job-ranking results over existing information retrieval approaches. We evaluate our system using various experiments on data from real-world recruitment documents.

Keywords

Information extraction; Semantic matching; Ontology; Relatedness; Semantic similarity; Knowledge graphs.

Citation: Brek, A., & Boufaida, Z. (2023). AnnoJOB: Semantic Annotation-Based System for Job Recommendation. *Acta Informatica Pragensia*, 12(2), 200–224. https://doi.org/10.18267/j.aip.204

Academic Editor: Stanislav Vojir, Prague University of Economics and Business, Czech Republic

1 Introduction

Recently, the e-recruitment phenomenon has been wide spread. The online job offers have increased, as well as the number of e-recruitment portals, such as Indeed (www.indeed.com), Glassdoor (www.glassdoor.com) and Monster (www.monster.com). Those portals have become the first destination for job seekers, where their searching technique is confined to using keywords such as job titles or skills, which returns a large number of results. These results are often irrelevant and poorly ranked, forcing the user to comb through the results, entailing a long and tedious process of checking each job description for relevance.

Resumes carry information unique to each job seeker, delivering a valuable base to the search for relevant jobs. Matching job descriptions with the candidate's resume enhances the job ranking process. Thus, the job description and resume content are of crucial importance. However, extracting information is a challenge since that information is unstructured, presented in different formats, as there are no defined templates for resumes or jobs contents, and contains different labels that refer to the same entities such as B.Sc, BS, Bachelors. Current information extraction systems, such as OpenCalais (www.opencalais.com) and Alchemy API (www.alchemyapi.com), are limited to extracting named entities based on traditional NLP techniques, which are insufficient for complex sentences, and most importantly, are unable to extract contextual entities. Semantic resources such as ontologies and knowledge graphs (KG) are proposed as a solution to structuring extracted entities by presenting contextual links between them to infer contextual entities (Bizer et al., 2005; Mochol et al., 2007). However, ensuring the knowledge quality and domain coverage in these resources is another problem.

While the information loss in the extraction process presents an issue, over-extracting entities also presents a problem, i.e., extracting entities that are not useful for the automated matching such as the applicant's name, graduation year, university and company name, etc.

Matching aims to assess how well a given resume fits a requested job or vice versa. In several works, the matching method is described as detecting lexical similarity between document entities, counting common words in both documents, or measuring a word's importance to a document within a collection. However, these methods are not adequate since there can be semantic heterogeneity in the texts. An example would be matching a job that requires "object-oriented programming" skills and a resume that offers the skill "java". Another problem is matching specific parts of documents ignoring the importance of other parts, such as comparing the required and the acquired skills, the required and acquired work experience without considering the context associated with the entities. For example, a job that requires "3 years of experience in Android" and a resume that offers "4 years of work experience as a developer of mobile applications". Those entities are semantically related, considering that work experience can mobilize a skill, and Android is a programming language for mobile applications. On the other hand, one might disregard the fact that some required entities can be inferred from other acquired entities, such as searching for a responsible candidate that has team spirit, the information "responsible" can be inferred from the fact that the candidate is "married", and their "team spirit" can be generated from their interest in "football".

Extracting contextual entities, presenting data from documents semantically, and using a rich source of information that is adaptable with data changing are the primary keys to enhancing job recommendation (Mishra et al., 2021; Brek et al., 2022).

This work aims to enhance job recommendation by considering the above issues. We assume that the quality and presentation format of the retrieved domain entities influence the matching process results. Therefore, we propose a job recommender system based on a semantic annotation approach to extract contextual entities and deliver a semantic presentation for the document data.

First, the offer/resume content is divided into blocks based on a keyword dictionary. Next, the Java Annotation Patterns Engine (JAPE) rules are applied to each block to extract domain entities intelligently. A domain ontology is developed to semantically inter-link the extracted entities and present them in a standard format using RDF triples. The matching module compares the RDF triples of the annotated jobs and resumes, looking for semantic similarity and relatedness between the extracted entities. The ontology and Wikidata are utilized as semantic background to assist the matching process to compute the relevance value between documents; the resume-relevant jobs are ranked based on this last.

The rest of this paper is organized as follows: Section 2 outlines the related work on using semantic technology to extract and match information in the e-recruitment field. Section 3 presents the proposed system (AnnoJob), giving a detailed description of the information extraction module and the semantic matching module. Section 4 provides results of the evaluation of the AnnoJob system. Finally, Section 5 presents the conclusions and future lines of work.

2 Literature Review

Online recruitment has been expanding notably, which is the reason behind a continuously increasing number of job descriptions online and the number of job seekers sending their resumes while searching for new opportunities. This massive amount of information increases the need to apply recommendation techniques to handle this information efficiently.

Recommender systems aim to suggest products that interest users by studying their behaviours (their product search) and their profiles (Ricci et al., 2015). As a recommender system, the job recommender system can retrieve a list of job positions that satisfy a job seeker's desire or a list of talented candidates that meet the requirement of a recruiter by using recommendation approaches. The mainstream approaches to recommender systems are classified into four categories: knowledge-based (KB), Content-Based Filtering (CBF), Collaborative Filtering (CF), and Hybrid approaches (Bobadilla et al.,2013; Wang et al.,2018). Based on several comparison studies in the e-recruitment field (Shaha et al.,2012; Tran et al.,2017; Dhameliya et al.,2019; Brek et al., 2020) , the Content-Based approach is more suitable for this context, according to the necessity of examining the content of domain documents to grant the best recommendation. This method consists of two primary modules: information extraction module and matching module.

2.1 Information extraction

Job seekers search for offers or companies looking for candidates on the internet. Whatever the purpose of the search, the information of the targets is mainly published as unstructured documents, which are difficult to process automatically. To address this issue, e-recruitment systems take advantage of information extraction techniques to extract the information required in matching resumes and offers.

Information extraction is an automatic technique for identifying entities from unstructured and/or semi-structured documents, intending to structure these documents and make them machine-readable. Based on previous research, the information extraction methods exploited in the job recommendation field can be divided into two categories: methods that exploit the document structure and text format, and others that exploit external resources.

Approaches that exploit document structure and format (Ciravegna et al., 2004; Chen et al., 2018) take advantage of the sentence syntax information, writing style, punctuation index, or even tags in HTML and XML files. These methods are difficult to adopt since it is impossible to know how many groups of documents follow the same template.

The primary purpose of semantic web techniques is to enrich web documents with machine-readable annotations, allowing automated document processing; semantic resources and techniques have recently been applied to guide the information extraction process (Martinez-Rodriguez et al., 2020).

Semantic methods mainly exploit semantic references such as knowledge graphs (KG) and domain ontologies to extract entities or relations from unstructured text. Domain ontologies have been used in different ways to extract and identify entities. Kumaran et al. (2013), Çelik et al. (2012), and Guo et al. (2016) use domain ontology as a semantic dictionary to extract information from resumes; the document content is segmented into sentences; next, the ontology concepts are used to detect domain entities. In the same way, Yahiaoui et al. (2006) and Karaa et al. (2011) propose a semantic annotation approach using a domain ontology to extract information from resumes and job offers. Using an ontology to extract entities takes advantage of exploiting semantic knowledge, which gives accurate results, but the ontology knowledge incompleteness problem still constrains it. To prevent this problem, Maree et al. (2019) utilized WordNet and YAGO3 (KG) to capture terms from resumes and offers after segmenting them into sentences and following extracting semantic and taxonomic relations (synonymy relation and hypernymy relation) to create a network that reflects the resume and the offer. The usage of YAGO3 and WordNet was insufficient, as certain concepts are absent; besides, the extracted relations were not always meaningful. Therefore, Ahmed Awan et al. (2019) propose a SAJ system that exploit both domain ontology and linked open data to annotate job offers. First, they use a domain-specific dictionary from JAPE rules to extract entities and then structure them in an RDF file based on the defined domain ontology. Next, an enrichment process is applied to enrich the ontology concepts with additional relevant labels from LOD.

Our information extraction approach takes into consideration the limits of the previous methods. We propose a semantic annotation approach that employs JAPE rules based on domain-specific dictionaries and sentence syntax to extract entities, then utilize a domain ontology to link the extracted entities semantically and yield contextual entities.

2.2 Semantic matching

Matching aims to determine how well a given resume fits a requested job by searching for the required information in the acquired information. Several techniques/approaches have been employed in this aspect, such as latent semantic analysis (LSA) (Lu et al., 2013), structured relevance models (SRM) (Yi et al., 2007), and fuzzy distance metrics (Daramola et al., 2010).

Semantic-based approaches achieve good, matching results in the e-recruitment domain. Martinez-Gil et al. (2016) define semantic matching as a function ($\mu 1 \times \mu 2 \Rightarrow R$) that associates the degree of correspondence for the entities $\mu 1$ and $\mu 2$ to a score $s \in R$ in the range [0, 1], where a score of 0 means no correspondence at all and 1 means total correspondence of the entities $\mu 1$ and $\mu 2$. The degree of correspondence of entities is estimated based on two different perspectives: using semantic similarity measures and semantic relatedness measures. Firstly, semantic similarity involves finding items that have a similar meaning. It is usually defined via the lexical relations of synonymy and hyponymy. Secondly, the basic idea of semantic relatedness defines items connected by any lexical or functional association. Dissimilar words can be semantically related.

From the description above, we can define the semantic matching process as evaluating the semantic similarity and relatedness between two documents based on semantic resources, such as domain ontologies and KG. In the literature, several methods are proposed to grant the best matching (resumes and offers) based on semantic references. Ontology-based methods can be broadly categorized as edge-counting approaches, feature-based approaches and information content-based approaches.

Edge-based approaches use these relationship paths to determine the similarity score between skills from resumes and offers. Let the path be define as $P\,(s1, s2) = e1, e2 \ldots en$, where e_i represents the edges connecting two skills along the path. The length of the path between two concepts is taken as the measure of similarity. Authors in Rácz et al. (2016; 2018), assume that having some skills imply that the applicant may have some other skills with certain probabilities. Therefore, they propose an edge-based ontology method to calculate the probability between skills using the subsumption relations.

Feature-based approaches assess the similarity between concepts as a function of their properties. They consider the degree of overlap between sets of ontological features (Balachander et al., 2018).

Content-based approaches, based on the amount of shared information between two terms, propose that the similarity between two concepts is based on the number of common subsumptions. Heggo et al., (2018) exploit a domain ontology to rank job offers based on extracting domain concepts from both offers and applicants; then, the ontology content is used to determinate how well the two concepts are related.

The semantics of the KG can enable delivering a meaningful description of the recommended items (Han et al., 2019) based on the semantic relation between users and items in the KG. In recent works, the KG has been used as a rich, structured and related source of data that can guide linking job seekers with relevant job offers. The resume and offer entities are presented with graph nodes, then multiple types of relations between the graph nodes are used to calculate the similarity between the documents **(**Upadhyay et al., 2021)**.**

In our system, we propose to exploit both domain ontology and KG to find semantic similarity and relatedness between resumes and job information. After reviewing existing approaches, it is clear that the following points require attention to improve e-recruitment systems.

- Information loss during the extraction process owing to absence of extraction rules adaptable to the context, text structure and format.
- Extraction of unuseful information using traditional methods such as NLP and named entity recognition, which extract entities blindly from text.
- Ignoring the relationships (hierarchical and associative) between the extracted entities leads to losing an essential information such as contextual entities.
- Unified semantic representation of document contents facilitates information accessibility, which improves the matching process.
- Inaccurate matching results based on lexical and semantic similarities between entities while ignoring semantic relatedness.
- Limited matching resources, as utilizing ontologies is confined to hierarchical relations and using KG is bounded by data quality.

3 AnnoJob system

The AnnoJob system represents a semantic annotation-based approach for job recommendation based on two main modules (detailed in Figure 1).

The information extraction module is led by the semantic annotation process; first, JAPE rules are defined using domain entity dictionaries, text structure and sentence syntax to extract entities from the text. Then, we develop a domain ontology to interlink the extracted entities and deliver them in a standard format using RDF triples. The output is an RDF file that represents the annotated resumes/offers; the RDF files that represent job descriptions are placed in the job database.

The semantic matching module intends to calculate the distance between the annotated resume/job descriptions by estimating the semantic similarity and relatedness between entities, utilizing semantic sources ontology and Wikidata. The results are a list of potential job offers for a candidate's resume.

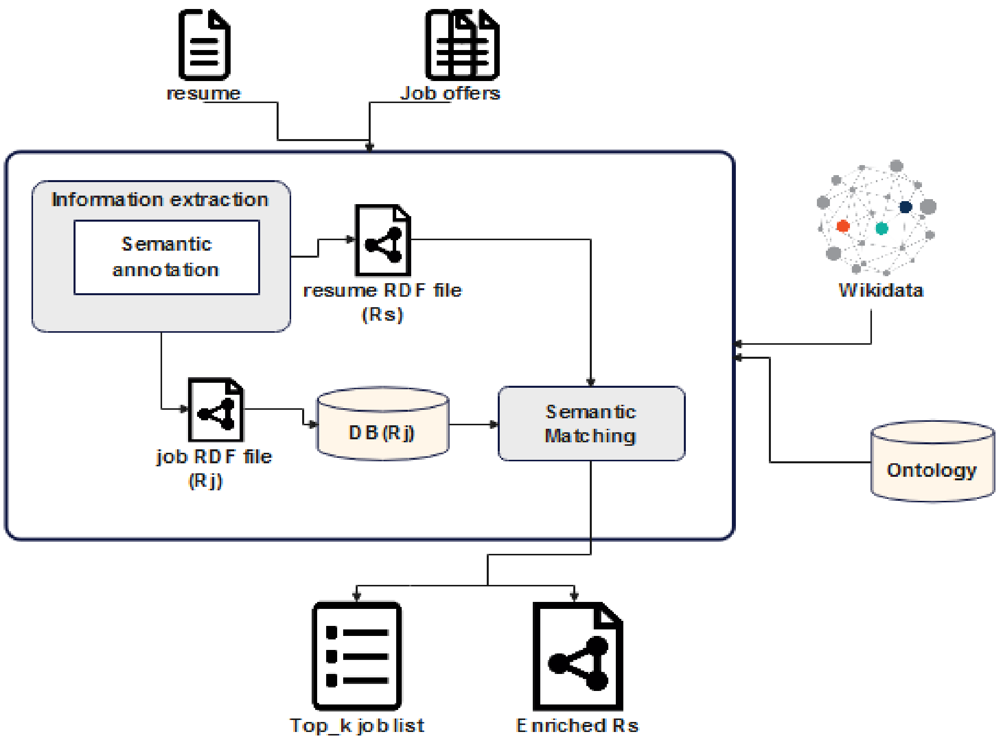


***Figure 1.** AnnoJob approach.*

3.1 Information extraction

E-recruitment documents are primarily published as unstructured text, which is difficult for the machine to process or exploit. The information extraction module intends to extract information and entities such as skills, diplomas, experience, job title and others essential for job recommendations. We propose to apply the semantic annotation technique to extract information and formally structure it, presenting its semantics. Our annotation approach utilizes the general annotation process (see Figure 2), which comprises three phases: pre-annotation, annotation and presentation. Our contribution resides in each phase.

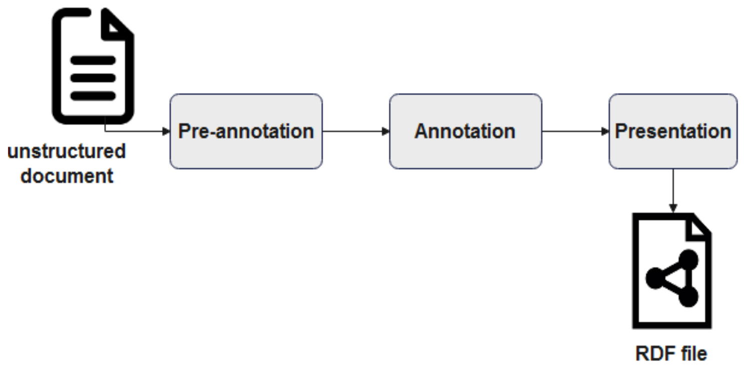


***Figure 2.** General annotation process.*

3.1.1 Pre-annotation phase

This phase assists text segmentation and entity extraction. The input is a set of job offers or resumes with different file types, such as doc, docx, and pdf. These files are processed by Tika (http://tika.apache.org) to get the raw text, where table layouts, font type and font colours are removed. The next step is structuring the extracted text in blocks based on a keyword dictionary that represents the label written at the beginning and defining the content of each part such as "skills", "education", "work experience" and "requirement". The block starts by identifying a label and ends with identifying the label of the next block. Some critical information does not begin with a label, like personal information written at the start of the resume; we identify it as the first block that starts from the first line of the text and ends at the first label. Afterwards, entities from blocks are extracted and structured in an XML file.

Resumes and job offers are published in different templates, which means some information may not be presented or may be labelled differently, such as "projects", "achievements", "interests", "responsibilities" and "benefits". To address this issue, we suggest using an XML skeleton (XSD file) to manage and standardize each resume/offer, emphasizing its functional parts.

Unlike the existing works, our extraction theory proposes extracting only the necessary entities for the matching module and ignoring others such as name, university name, phone number, email, graduation year, or company name. Therefore, we developed extracting rules using JAPE grammar. JAPE grammar uses features for building pattern/action rules. The feature set contains aspects such as POS tags, a dictionary of entities presented in gazetteer lists, and punctuation. The block is defined based on the tag names of the XML skeleton, and then the JAPE rules are selected and applied to the block's contents to extract entities and write XML element values (tag value and attribute value; see Figure 3).

All the extracted entities are normalized before being added to the XML file, where entities are written in small letters and an underscore symbol is added to link the words together (if the entity consists of several words). For example, the entity "mobile app developer" is replaced by its normalized form "mobile_app_developer". Moreover, in the personal information, the candidate’s age is inferred from the delivered date of birth.

The result of this phase is an XML file containing the extracted entities structured based on the XML skeleton. Figure 4 presents an example of a resume XML file.

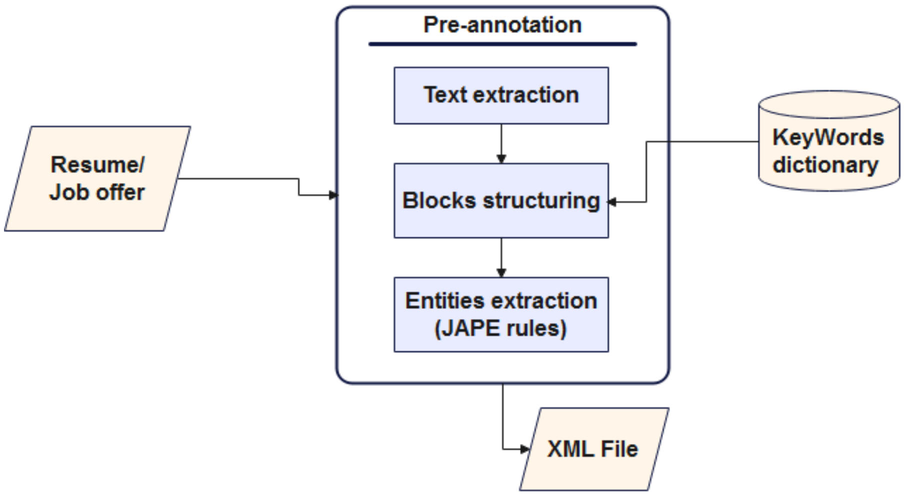
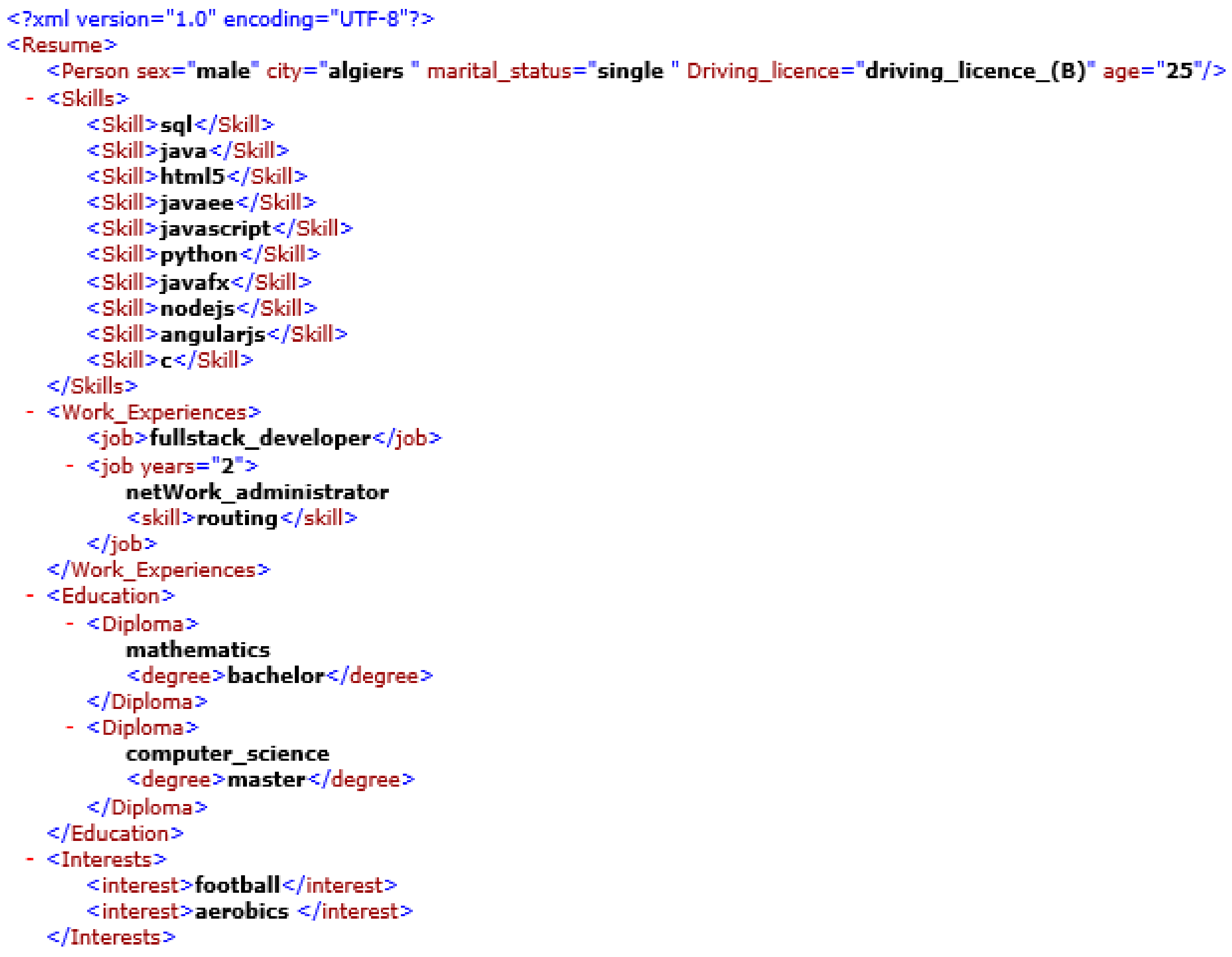


Figure 3. *Pre-annotation phase.*



```
<?xml version="1.0" encoding="UTF-8"?>
<Resume>
    <Person sex="male" city="algiers " marital_status="single " Driving_licence="driving_licence_(B)" age="25"/>
  - <Skills>
        <Skill>sql</Skill>
        <Skill>java</Skill>
        <Skill>html5</Skill>
        <Skill>javaee</Skill>
        <Skill>javascript</Skill>
        <Skill>python</Skill>
        <Skill>javafx</Skill>
        <Skill>nodejs</Skill>
        <Skill>angularjs</Skill>
        <Skill>c</Skill>
    </Skills>
  - <Work_Experiences>
        <job>fullstack_developer</job>
      - <job years="2">
            netWork_administrator
            <skill>routing</skill>
        </job>
    </Work_Experiences>
  - <Education>
      - <Diploma>
            mathematics
            <degree>bachelor</degree>
        </Diploma>
      - <Diploma>
            computer_science
            <degree>master</degree>
        </Diploma>
    </Education>
  - <Interests>
        <interest>football</interest>
        <interest>aerobics </interest>
    </Interests>
```

Figure 4. *XML file (resume example).*

3.1.2 Annotation phase

In this phase, a domain ontology is utilized to annotate the obtained XML file by presenting the ontology elements relevant to the XML elements and establishing semantic relations (both hierarchical and associative) among extracted entities, delivering the extracted information as a knowledge graph.

As shown in Figure 5, the developed ontology presents the most essential and common parts of resumes and job offers. The primary purpose of using this ontology is to achieve a unified presentation of the extracted information and build a knowledge graph based on its elements.

The concept "person" with its attributes represents the personal information in the resume, which is the personal information required in job offers such as age, sex, driving license, or address when demanding that an applicant resides in an area near the job location. Furthermore, the "job" concept represents the work experience in the resume or the job offered in the job descriptions, and the candidate that occupies a job position, or works on a project is gaining experience in specific skills, which is presented with the relation "has-experience".

Based on the cisco company website (https://www.cisco.com), being certified makes a considerable difference in a job search. 99% of employers rely on certifications to make decisions; they consider a certified candidate to have more knowledge than others. In other words, having the certification proves the candidate's level in some skill or language. This is represented by the concept "certification", related to the concepts "skill" and "language" with the relation "proves". Besides skills, diplomas, certifications, and job experience, considering "assets" and "interests" is essential in resumes and offers. For example, in a job offer, the asset "team spirit" is required; this concept may not be directly mentioned in a resume but can be deduced from the interest in football or work experience in a team. The concept "document" represents the annotated document, which is defined with an "id", a "title" and "type" that defines whether the document is a resume or a job offer.

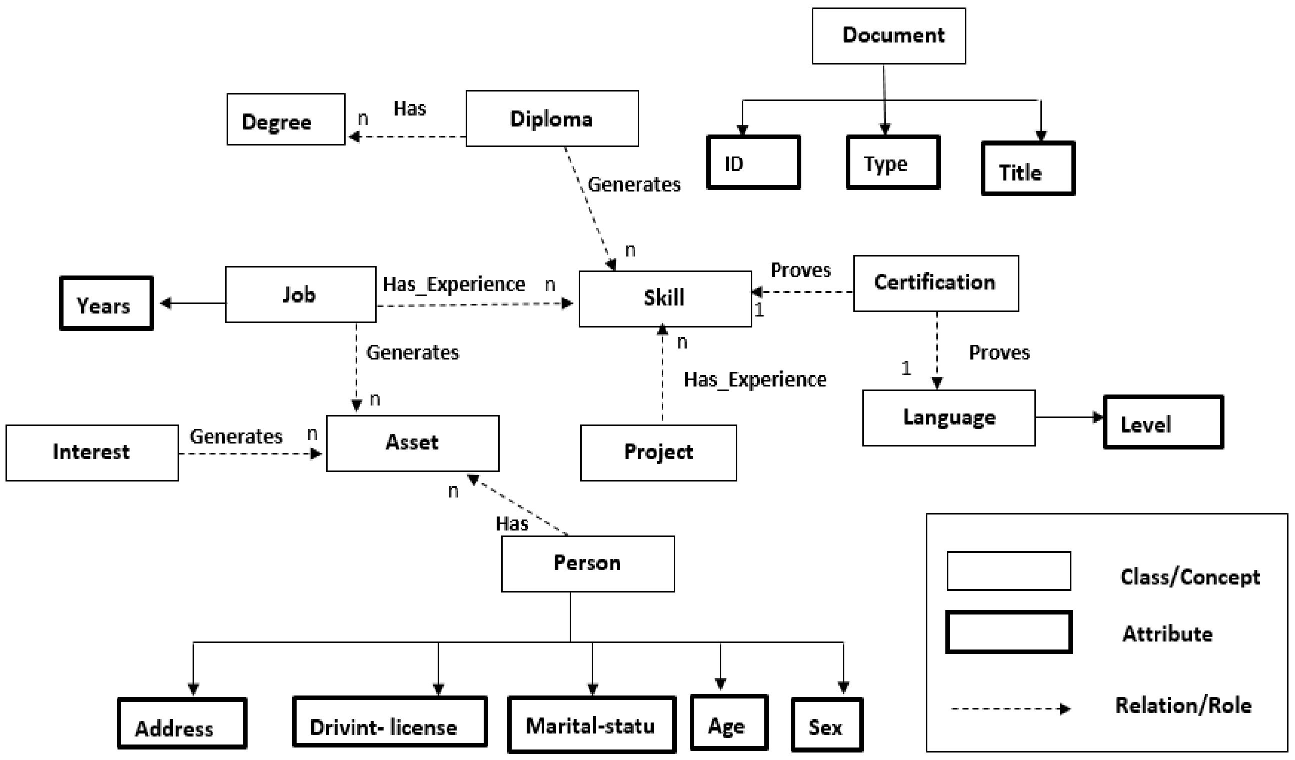


Figure 5. *The domain ontology.*

To annotate the XML file with the ontology elements, we exploit the XML structure to define rules that select the appropriate ontological element for the XML element (see Table 1). First, we select the XML tag name and search for the concept that matches that name. Then, we search for the tag value in the instances of that concept. Once the instance is found, the tag value is annotated using the instance URI. Otherwise, the tag value is added to the ontology as a new concept instance. Annotated entities are also forwarded to the inter-connection stage, where we add links between entities based on the ontology elements to produce a knowledge graph. Those links represent relations and information that may not be explicitly defined in the documents. For example, adding a link between certification and language proves the required language level. Adding a link between "TCF C2" and "French" presents the required information, such as "fluent in French".

Table 1. *XML annotation rules.*

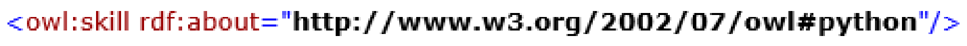
XML elements	Ontology elements
Tag name	Concept name
Tag value	Instance
Attribute name	Attribute name
Attribute value	Attribute value
Hierarchical relation	Object property

3.1.3 Presentation phase

The annotation can be supplied in various formats, including JSON, XML and RDF. The structure of the extracted knowledge may be seen from the relations between entities and their labels and, attributes, and between entities themselves. Therefore, we deliver our annotation as RDF triples. The RDF triple is presented in three elements: subject, predicate and object. Table 2 shows the possible cases of annotations.

Both resumes and job offers are presented in RDF triples based on the provided description, which grants a unified representation of the extracted information. This annotation is exploited in the matching module.

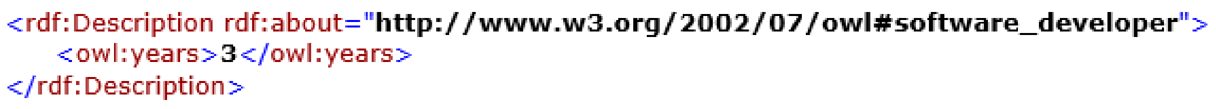
Case 1: The RDF triple represents the relation between the extracted entity (the instance URI that represents the XML tag value) and the annotation (the concept that represents the XML tag name; see Figure 6).



```
<owl:skill rdf:about="http://www.w3.org/2002/07/owl#python"/>
```

Figure 6. *RDF triple case 1.*

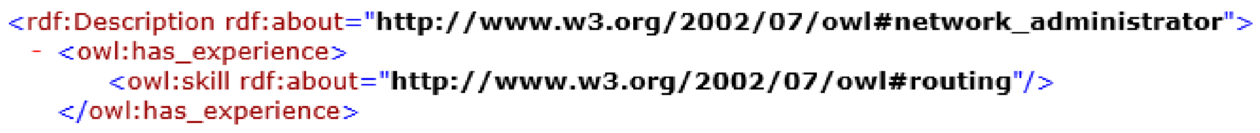
Case 2: The RDF triple represents the relation between the extracted entity and the attributes; the attribute name defines the relation between the instance and the attribute value (see Figure 7).



```
<rdf:Description rdf:about="http://www.w3.org/2002/07/owl#software_developer">
    <owl:years>3</owl:years>
</rdf:Description>
```

Figure 7. *RDF triple case 2.*

Case 3: The RDF triple represents the relation between two extracted entities, defined by the object property between two instances (see Figure 8).



```
<rdf:Description rdf:about="http://www.w3.org/2002/07/owl#network_administrator">
  - <owl:has_experience>
        <owl:skill rdf:about="http://www.w3.org/2002/07/owl#routing"/>
    </owl:has_experience>
```

Figure 8. *RDF triple case 3.*

3.2 Semantic matching

Our matching theory is based on estimating the distance between the two RDF files that represent the annotated resume and job description. The distance is defined by the semantic similarity and the semantic relatedness between the RDF triples exploiting the domain ontology and Wikidata as semantic references. The result of this module is a list of the most relevant job offers for a resume, defined by the distance value, ranging from 1 for perfect matching and 0 for no matching. Figure 9 illustrates the proposed semantic matching process.

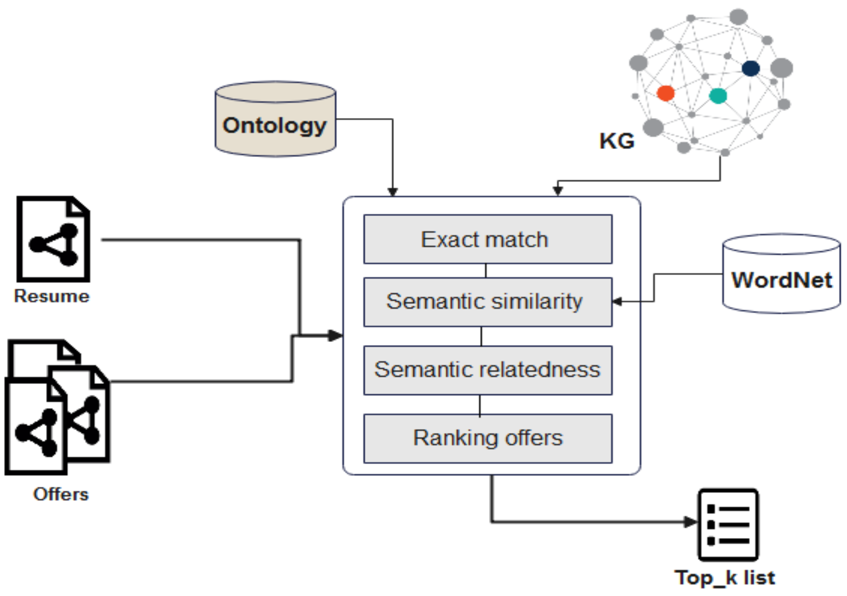


Figure 9. *Semantic matching process.*

The main idea consists of finding the exact match between the resume and offer RDF files by searching for each required triple from a job offer in the set of acquired triples from a candidate's resume. In the simplest case, if the required triple and the acquired triple are the same, the matching value is 1. If they are different, we can check whether the required triple is similar or related to the acquired triple based on the semantic similarity measure and semantic resources. If this is the case, the matching value is 0.5; otherwise, the matching value is 0. The distance is defined by the sum of the matching value scaled by the number of required triples. The distance is measured as follows:

$$Dis\ (Rs, Rj) = \frac{\sum_{\substack{1\leq i\leq n \\ 1\leq y\leq m}} match(Rji,Rsy)}{n} \qquad (1)$$

Where n is the number of the required triples (*Rj*) and m is the number of the acquired triples (*Rs*). The *match(Rj*$_i$*,Rs*$_y$*)* value is defined as :

$$match(Rji\ , Rsy)\ = \begin{cases} 1, if\ Rji = Rsy \\ 0.5, if\ rel(Rji, Rsy) \\ 0, \quad otherwise \end{cases} \qquad (2)$$

Where *rel(Rj*$_i$*,Rs*$_y$*)* is a Boolean function that returns the value “true” if there is semantic similarity or relatedness between the required and the acquired triple, and returns “false” otherwise. If the required triple does match any acquired triples, the function is used to verify their semantic similarity or relatedness. We believe that some required information is not explicitly mentioned in the resumes and can be extracted from other entities or relations in the resumes using a domain ontology as a guide.

The RDF triples represent three types of annotation based on the extracted information, which can represent an entity, attribute value of an entity, or relation between two entities. Therefore, we compare the triples based on their elements and content using the subject, object and predicate. In the first case where the triple predicate is "rdf:type", we compare the required and acquired triple objects with the same triple subject or related subjects based on the ontology using the sparql query "Q1" in listing 1.

Listing 1. *SPARQL query for related subject extraction over the ontology.*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?s
WHERE{{?prop  rdfs:domain ?s; rdfs:range "+sub+".}
union{?prop  rdfs:domain  "+sub+" ; rdfs:range ?s.}}
```

Algorithm 1 presents the general process of semantic matching, whereas the "rel function" is detailed in Algorithm 2.

Algorithm 1. *Semantic matching algorithm.*

```
1:input Resume RDF triples of $Rs = \{rs_1, rs_2, rs_3, ... rs_i\}$, Job RDF triples of $Rj = \{rj_1, rj_2, rj_3, ... rj_i\}$.
2:output real distance.
3: match ←0
4:rpns ← false
5:lst ← $\phi$
6: For $rj_i \in Rj$ do
 7: if $Rs.contains(rj_i)$ then
 8: $match \leftarrow match$ + 1
 9: else
   10: pred ← $rj_i$.getPredicate()
   11: sub ← $rj_i$.getSubject()
   12: obj ← $rj_i$.getObject()
   13: Lst ← Q1(sub)  //Q1 SPARQL query to get all related classes to sub from ontology
   14:for $rs_i \in$ Rs do
     15: preds ← $rs_i$.getPredicate()
     16: subs ← $rs_i$.getSubject()
     17: objs ← $rs_i$.getObject()
     18: if pred = preds then
       19: if pred = RDF.type then
          20:if (sub = subs||subs ∈ Lst) then
           21:rpn ← rel(obj, objs)
           22:if rpn = true then
             23:match ← match + 0.5
             24:end if
          25:End if
    26:else // the case where the triples predicate not rdf:type
     27:if (sub = subs||Q2(sub, subs) = true) then //Q2 SPARQL query to find relation between the subjects in ontology
       28:if obj = objs then
        29:match ← match + 1
    30:end if
   31:end if
   32:end for
 33:end if
34:end for
35:distance ← match/(Rj.size())
36:Return distance.
```

Algorithm 2. *rel function algorithm.*

```
1:input obj, objs.
2:output Boolean result.
3:val ← 0
4:result ← false
5:rlt ← false
6:val ← sim(obj, objs) // find the semantic similarity
 7:if val > 0.5 then
 8:result ← true
 9:else
 10:rlt ← Q2(obj, objs) //Q2 SPARQL query to find relation between the      subjects
in ontology
 11:if rlt = true then
   12:result ← true
   13:else
     14:rlt ← Q3(obj, objs) //Q3 SPARQL query to find relation in Wikidata
     15:if rlt = true then
       16:result ← true
     17:end if
    18:end if
  19:end
20:Return result.
```

3.2.1 Semantic similarity

In the field of information technology, to compare entities, we face two significant problems: entities closer in terms of writing, such as "java, java fx, java EE or angular 8, angular js", and entities that are differently written but related based on their definition such as "network director" and "telecommunication manager".

The semantic similarity step aims to estimate the similarity between the required triple (Rj) and the acquired triples (Rs) with the same "rdf:type" value. This done based on two similarity measures: string similarity using Levenshtein distance and semantic similarity using the Leacock and Chodorow (LCH) measure (Fellbaum & Miller, 1998) over the WordNet dictionary.

$$Sim(Rj, Rs) = Lev\,(Rj, Rs) + Lch\,(Rj, Rs) \qquad (3)$$

The similarity score is the sum of the two-measure value (see Equation 3). First, we exploit the *Levenshtein* distance to capture similarities between the two strings (Rj and Rs); the distance is scaled by the length of the longest string as follows:

$$Lev\,(Rj, Rs) = \frac{(longerLength - Levensh\,(Rj,Rs))}{longerLength} \qquad (4)$$

The distance score ranges between 1 and 0: 1 for totally similar and 0 for no similarity.

Next, to handle the entities that are differently written but related based on their definition such as "network director" and "telecommunication manager", the entities are submitted to the LCH measure. The LCH method counts the number of edges between two words in the WordNet “is-a” hierarchy. The value is then scaled by the maximum depth of the WordNet “is-a” hierarchy. A similarity value is obtained by taking the negative log of this scaled value. The LCH method is exploited via WS4J API. The formulation is as follows:

$$Lch(Rj, Rs) = max\left(-log\,\frac{ShortestLen(Rj,Rs)}{2*TaxonomyDepth}\right) \qquad (5)$$

The similarity score is 0 for no similarity and greater than 0 if there is a similarity.

In this step, we only search for the semantic similarity between the triple resources with the same "rdf:type" value. So, for example, in the required triple where the resource "javaEE" has "rdf:type" value "skill", we compare "javaEE" with all acquired resources that have the "rdf:type" value "skill". Our experiments determined the similarity threshold by "sim>0.5", where the function gives more coherent results.

3.2.2 Semantic relatedness

As mentioned above, we employ ontology and Wikidata to identify relatedness between triple resources. Our ontology is developed to represent the e-recruitment domain based on relations between domain concepts and instances that deliver real-world information. The ontology is used as a guide for matching by employing the relations between concepts. Otherwise, the ontology may not be rich enough to determine relatedness between instances. Therefore, we need a rich KG to support the matching and enrich the ontology. Based on comparative studies (Färber et al., 2017; Färber et al., 2018), Wikidata is a more suitable knowledge graph for this context. Since it is known as an instance KG rather than a schema KG that contains the highest instance number, it presents general information that is continuously updated.

Most of the relevant approaches exploit Ontology Edges to define relatedness between entities from resumes and offers; for example, Guo et al. (2016) exploit a skill ontology to define the relatedness between skills such as "java" and "python". However, from our point of view, if employers search for candidates with a "java" skill, recommending a candidate with a "python" skill will not be helpful even if they are related based on the defined ontology since the required information in this field is critical and strict. Therefore, we implicitly search for the required information in the acquired information using semantic relatedness.

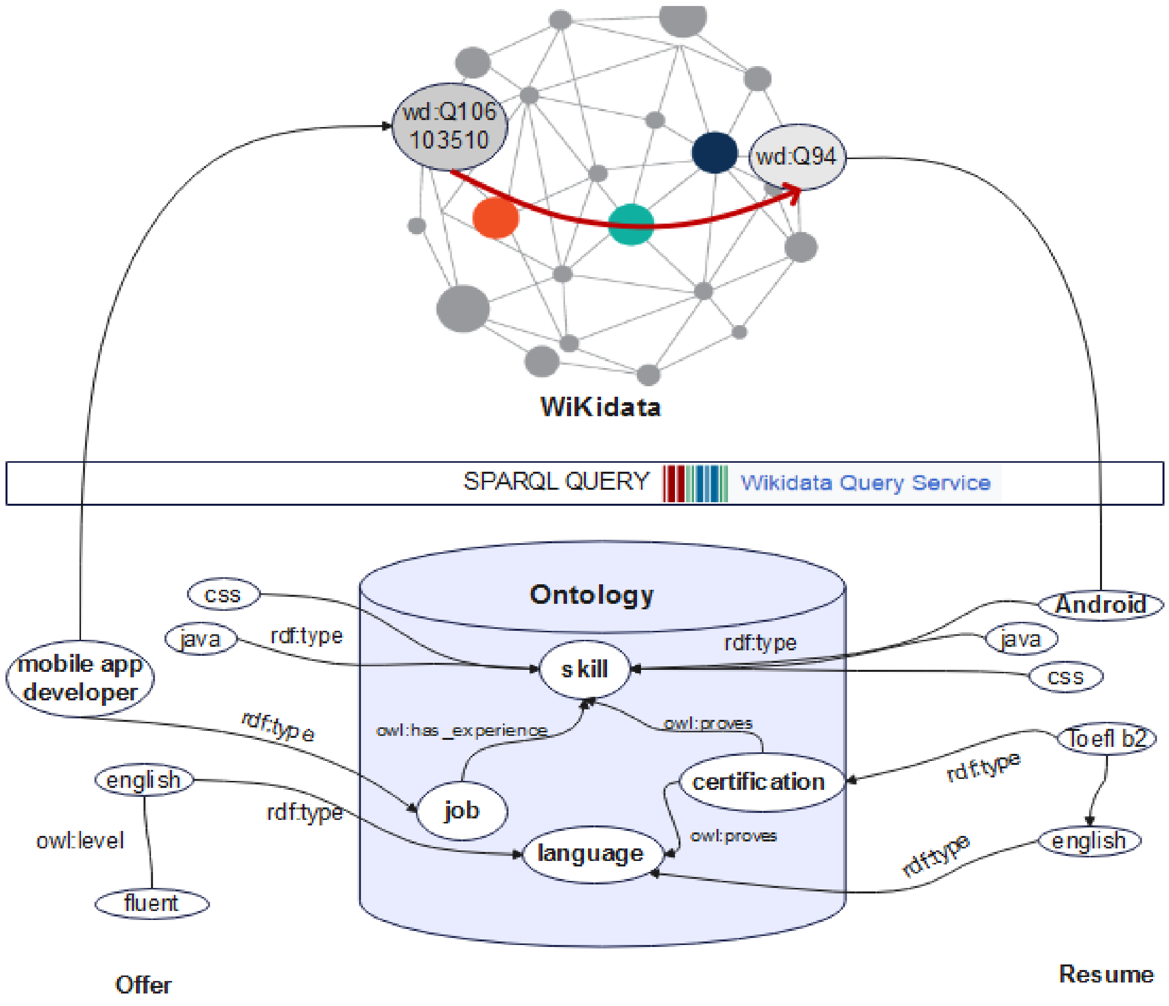


Figure 10. *Example of finding relatedness between an annotated offer and resume using ontology and Wikidata.*

Figure 11 illustrates an example of a given offer and resume annotated with the domain ontology. From the offer, the entity "mobile app developer" is annotated with the concept "job".

To find relatedness, first, we need to determine what entities from the resume are to be compared with the required entity. From algorithm 1, Q1 (Listing 1) is a SARQL query used over the ontology to search for concepts related to "job". In this example, "job" is related to "skill". Therefore we can search for relatedness between the required entity "mobile app developer" and all the acquired skills "java, CSS, android" using first Q2 (Listing 2) over the ontology and then Q3 (Listing 3) on Wikidata.

Listing 2. *SPARQL query to retrieve the relatedness over the ontology.*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
ASK WHERE {
?prop rdfs:domain "+obj+"; rdfs:range "+objs+".}
```

Generally, to search for a relation between two or more entities over a KG, we can use the relation label if it is known, or the path syntax in the SPARQL query. In our case, any direct relation between the two given entities implies that those entities are related. As with any KG, Wikidata may miss some relations; therefore, we exploit the description of the entities to find related entities. If an entity appears in the description of the other entity,it means those two are related. In our example, “Android” in Wikidata is described as an “open-source operating system for mobile devices” the appearance of the word “mobile” proves that “Android” is related to “mobile app developer”.

Listing 3. *SPARQL query to retrieve the relatedness from Wikidata.*

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX v: <http://www.wikidata.org/prop/statement/>
Ask WHERE {SERVICE wikibase:mwapi
{bd:serviceParam wikibase:api "EntitySearch".
 bd:serviceParam wikibase:endpoint "www.wikidata.org".
 bd:serviceParam mwapi:search "+obj+".
 bd:serviceParam mwapi:language "en".
 ?r wikibase:apiOutputItem mwapi:item.}
 {bd:serviceParam wikibase:api "EntitySearch".
 bd:serviceParam wikibase:endpoint "www.wikidata.org".
 bd:serviceParam mwapi:search "+objs+".
 bd:serviceParam mwapi:language "en".
 ?s wikibase:apiOutputItem mwapi:item.}
 ?r ?PRP1 ?s.
 ?s ?PRP2 ?r.
 ?s rdfs:label ?s_label
 FILTER(CONTAINS(LCASE(?s_label), "+?obj+"@en)).
 ?r rdfs:label ?r_label
 FILTER(CONTAINS(LCASE(?r_label), "+?objs+"@en)).
 OPTIONAL{?s schema:description ?o FILTER
 (CONTAINS(LCASE(?o),"+?objs+"@en) &&
  CONTAINS(LCASE(?o), "+?obj+"@en))}
  OPTIONAL {?r schema:description
  ?t FILTER (CONTAINS(LCASE(?t), "+?obj+"@en)
          && CONTAINS(LCASE(?t), "+?objs+"@en))}}
```

This SPARQL query returns the value "true" if there is relatedness between the two objects (obj) and (objs). The relatedness between entities is measured either by finding a relation (?PRP1, ?PRP2) between the two entities, or by the exitence of one of the entities in the other entity's description (schema:description). To normalize an entity, an underscore symbol is added to link the words together (if the entity consists of several words). For example, the entity "mobile app developer” is replaced by its normalized form “mobile_app_developer”. As known, every entity in Wikidata is identified with a QID, which is the unique identifier of a data item on Wikidata. In our case, the query is automatically generated in the matching phase, as it is hard to know the QID for each entity. Therefore, we exploit "SERVICE wikibase:mwapi" in the query to search for the QID of the given entity.

In the other case, where the triple predicate is different from "rdf:type", the required triple is compared with the acquired triples with the same predicates. If the triple subjects are the same or related, we compare the objects.

Figure 10 represents a required triple "having 2 years’ experience in java". To match this triple, we search for triples with the same predicate. In our example, this is "year". Next, we compare the two subjects; here, they are not the same but are related. Therefore, we compare the two objects. If they are matched, the match value is "1".

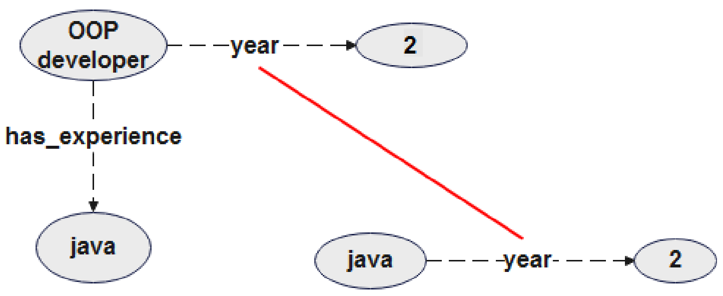


Figure 11. *Example of matching two triples.*

In a particular case, we need to exploit annotation from the ontology to compare the triples. For example, let us take the triples that define the educational information; the triple represents the relation between two instances: the diploma and the degree. First, we compare the required and the acquired diploma instances employing the relation (related to) from the ontology. Next, to match the required and the acquired degree, such as bachelor's and master's, in the ontology, we assign the annotation "isDefinedBy" with the attribute "Value" to define an integer value for each degree to check whether a candidate is under-qualified compared to the academic qualification requirement (see Figure 12). (e.g., a master's degree is assigned a value of 3, while a bachelor's degree is assigned a value of 1). If the required degree value is higher than the acquired one, the function *rel* takes the value "true".

```
<NamedIndividual rdf:about="&owl;bachelor">
    <rdfs:isDefinedBy>
        <rdf:Description>
            <value rdf:datatype="&xsd;integer">1</value>
        </rdf:Description>
    </rdfs:isDefinedBy>
</NamedIndividual>
```

Figure 12. *Degree annotation.*

Once the resume is compared to all job offers from the job base, the offers are ranked based on the distance value and delivered in a list. Moreover, we enrich the annotated resume with the job title of the most relevant offer, which will be the key to the future job search.

4 Experimental Evaluation

Our system mainly uses the semantic annotation approach to extract entities and exploit the obtained semantic annotation to match resumes and jobs. We assess the performance of the AnnoJob system by studying the results of the two main components of the system: information extraction and semantic matching.

4.1 Implementation aspect

The AnnoJob system is a java web application. The implementation uses Gate embedded, Protégé OWL API to work with the OWL ontologies, and the Jena API to manipulate the different constituents of the ontology (classes, relations, instances, data type properties, etc.).

Figure 13 illustrates the AnnoJob component diagram, including the two main components (information extraction and semantic matching). The aim is to upgrade, maintain, and enhance each component while keeping the overall system code unaffected. This architecture also guarantees flexibility, enabling it simple to inclusion of new components, thus allowing system expansion.

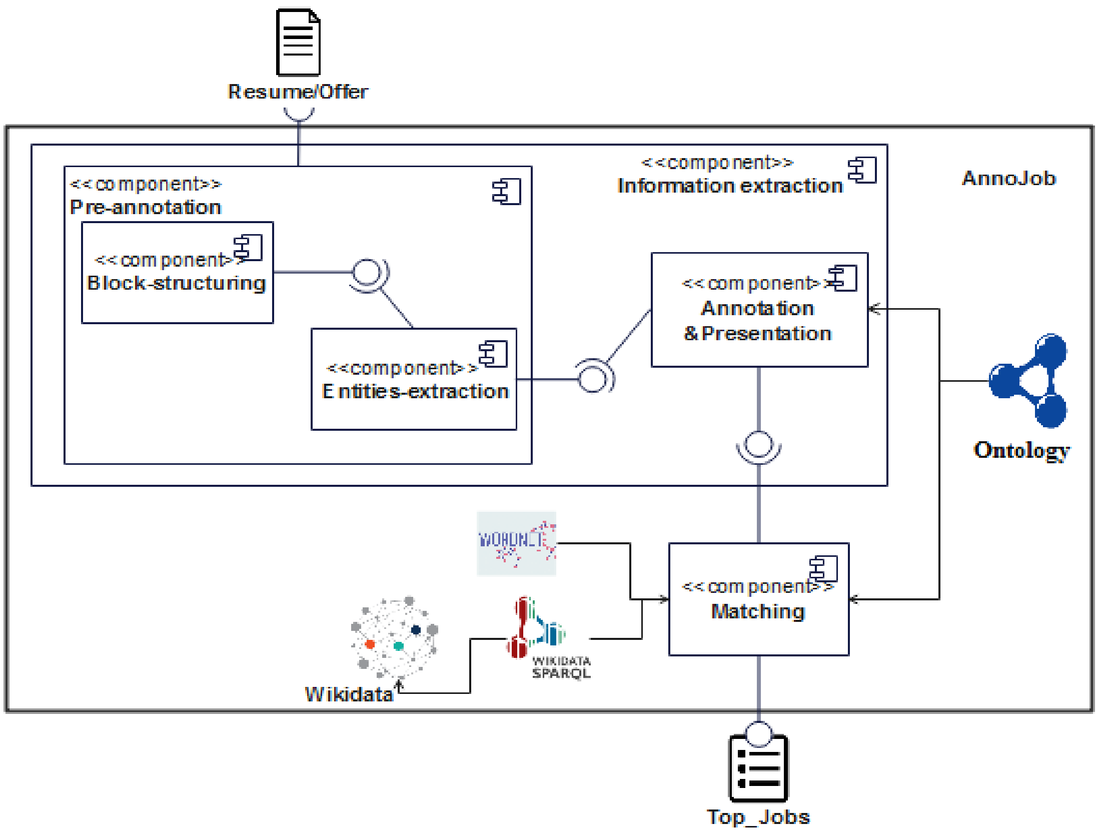


Figure 13. *Component diagram of AnnoJob.*

Figure 14 presents the AnnoJob system interface. Once the user uploads his resume, it is annotated and compared with jobs from the database to select the most relevant jobs.

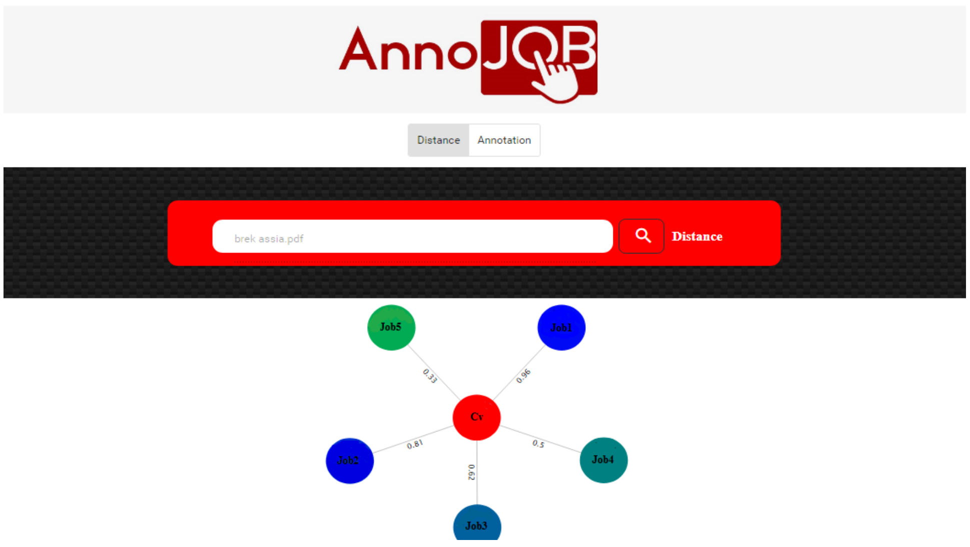


Figure 14. *Top_K ranked offers using AnnoJob.*

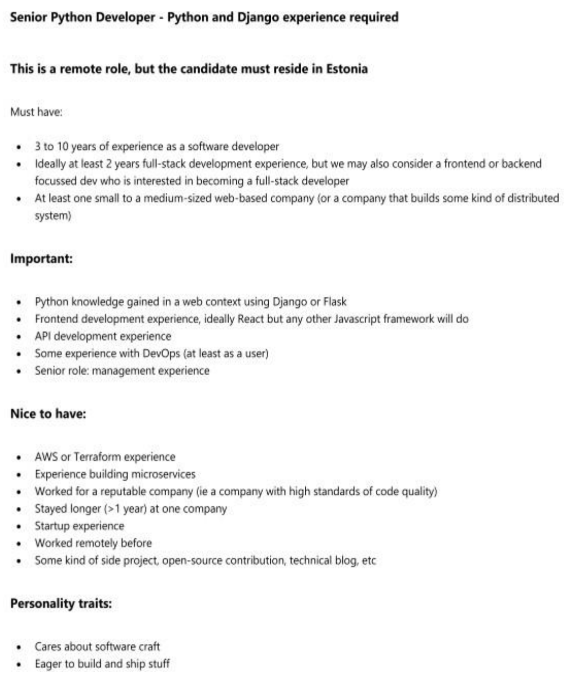
4.2 Dataset acquisition

To assess the performance of our system, we evaluated it on real-world data. As there is no standard dataset available for job descriptions or resumes, we gathered a dataset of 1000 job descriptions downloaded from Indeed (www.indeed.com), Kaggle (www.kaggle.com), data.world and app.datastock.shop. We collected 100 resumes from LinkedIn (www.linkedin.com). Since the user's profile and resume are confidential information on LinkedIn, we published a request on our LinkedIn profile to collect resumes for the experiment. The datasets collected belong mainly to information technology and other domains such as healthcare and safety, sales and marketing, architecture and engineering, management, and executive and customer care.

Figures 15 and 16 present an example of the collected job offers from Indeed and resumes from LinkedIn, where the documents are in pdf and dox format. On the other hand, the collected data from other websites were in csv files or txt files where the offer description was limited to job title and required skills.

Table 2. *Statistics of resumes/offers in various job categories collected.*

Job categories	**Count**
Back-end Developer	100
Front-end Developer	150
Full-stack Developer	215
Cryptocurrency/Blockchain	80
Cyber Security/Network	230
Data science	190
Other	140
Total	1100



Senior Python Developer - Python and Django experience required

This is a remote role, but the candidate must reside in Estonia

Must have:

- 3 to 10 years of experience as a software developer
- Ideally at least 2 years full-stack development experience, but we may also consider a frontend or backend focussed dev who is interested in becoming a full-stack developer
- At least one small to a medium-sized web-based company (or a company that builds some kind of distributed system)

Important:

- Python knowledge gained in a web context using Django or Flask
- Frontend development experience, ideally React but any other Javascript framework will do
- API development experience
- Some experience with DevOps (at least as a user)
- Senior role: management experience

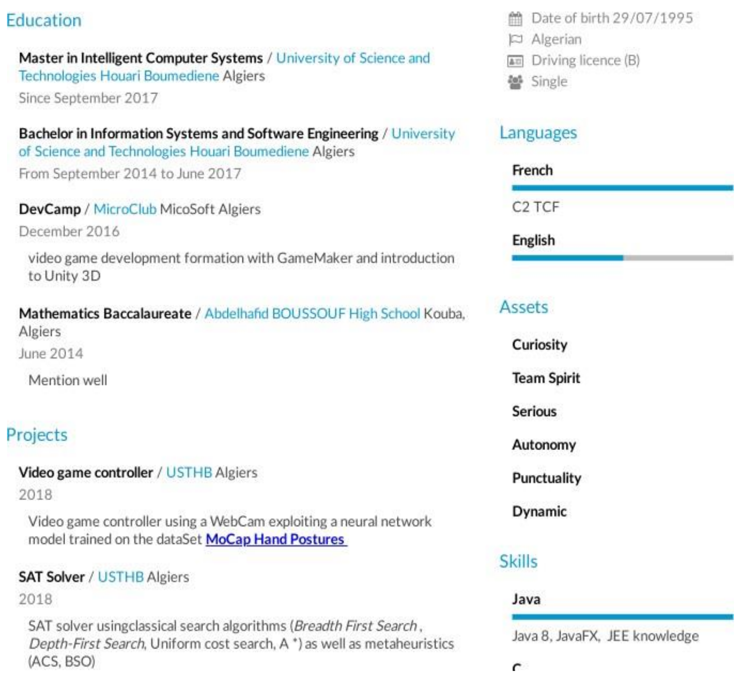
Nice to have:

- AWS or Terraform experience
- Experience building microservices
- Worked for a reputable company (ie a company with high standards of code quality)
- Stayed longer (>1 year) at one company
- Startup experience
- Worked remotely before
- Some kind of side project, open-source contribution, technical blog, etc

Personality traits:

- Cares about software craft
- Eager to build and ship stuff

Figure 15. *Example of job description.*



Education

Master in Intelligent Computer Systems / University of Science and Technologies Houari Boumediene Algiers
Since September 2017

Bachelor in Information Systems and Software Engineering / University of Science and Technologies Houari Boumediene Algiers
From September 2014 to June 2017

DevCamp / MicroClub MicoSoft Algiers
December 2016
video game development formation with GameMaker and introduction to Unity 3D

Mathematics Baccalaureate / Abdelhafid BOUSSOUF High School Kouba, Algiers
June 2014
Mention well

Projects

Video game controller / USTHB Algiers
2018
Video game controller using a WebCam exploiting a neural network model trained on the dataSet MoCap Hand Postures

SAT Solver / USTHB Algiers
2018
SAT solver usingclassical search algorithms (*Breadth First Search* , *Depth-First Search*, Uniform cost search, A *) as well as metaheuristics (ACS, BSO)

Date of birth 29/07/1995
Algerian
Driving licence (B)
Single

Languages

French
C2 TCF
English

Assets

Curiosity
Team Spirit
Serious
Autonomy
Punctuality
Dynamic

Skills

Java
Java 8, JavaFX, JEE knowledge

Figure 16. *Example of resume.*

4.3 Information extraction assessment

This module uses the semantic annotation technique to extract and structure document information. To evaluate the semantic annotation results, we used the measure of precision and recall to report how well this technique can identify information from a document. F-measure is used as a weighted harmonic mean of precision and recall. We denote the relevant entities in the resume/offer as $E = \{e_1, e_2, \dots e_n\}$ and the retrieved entities as $\hat{E} = \{\hat{e}_1, \hat{e}_2, \dots \hat{e}_n\}$.

$$Recall = \frac{(E \cap \hat{E})}{E} \tag{6}$$

$$Precision = \frac{E \cap \hat{E}}{\hat{E}} \tag{7}$$

$$F - 1 = \frac{2 * precision * recall}{precision + recall} \tag{8}$$

Table 3. *Results of information extraction assessment.*

	Job offers			Resumes		
	Recall	**Precision**	**F-1**	**Recall**	**Precision**	**F-1**
Skills	0.880	0.911	0.895	0.948	0.961	0.954
Education	0.851	0.832	0.841	0.902	0.970	0.934
Work experience	0.792	0.810	0.800	0.843	0.830	0.836
Personal information	0.723	0.734	0.728	0.972	0.987	0.979
Assets/Interests	0.761	0.774	0.767	0.820	0.846	0.832

Table 3 shows the semantic annotation approach results for extraction of required/acquired entities from documents. Exploiting entity dictionaries, punctuation and sentence syntax in the extraction rules (JAPE rules) improves the extraction module and gives fascinating results. Based on the table, we can see that the extraction results for resumes are better than those for job offers due to how the information is written. The resume content presented in small sample sentences and a standard format differs from the job description, where sentences are more complicated, making extracting all correct entities more difficult. However, based on these results, we notice that our approach gives stable and balanced results in annotating different information from different parts of the text. The recall results prove that the system is able to extract most of the relevant entities and the precision results show that most of the retrieved entities are relevant.

4.4 Matching evaluation

In this part, we not only analyse the matching module but also evaluate the effectiveness of the semantic annotation in the matching process. Our theory implies that the unified presentation of the resumes/offers and the exploitation of semantic sources (domain ontology, Wikidata), which enhance and facilitate the matching process.

As described in the past section, we defined a semantic similarity measure in Equation (3) to verify whether two given entities are similar, where the similarity score ranges between 0 and 1. To improve the results of the equation, we needed to determine a threshold where the equation gives more relevant and fewer irrelevant results. We exploit the receiver operating characteristic curve (ROC curve), mainly used to evaluate the performance of a recommender system based on determining the best threshold to recommend the maximum relevant items and minimum irrelevant items. Based on our experiments and studies, the similarity score threshold is defined as 0.5; If the similarity score between two entities is higher than 0.5, the two entities are correctly classified as related.

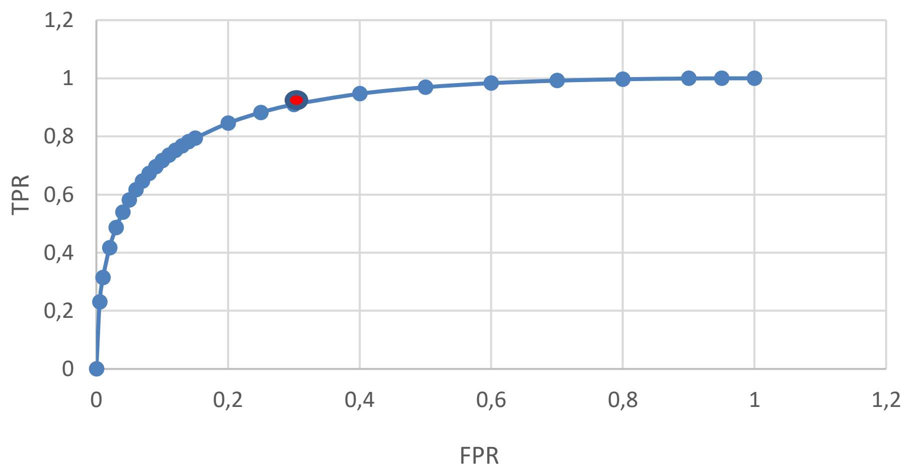


Figure 17. *ROC curve.*

As shown in Figure 17, we calculated the true positive rate (TPR; sensitivity) and the false positive rate (FPR) using different threshold options. The TPR represents what percentage we classified correctly and FPR represents what percentage we miss-classified based on the given threshold. The red point represents the TPR and FPR values where the similarity threshold value = 0.5. At this point, we got the maximum relevant results and the minimum irrelevant results.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad (9)$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{10}$$

To evaluate the quality of the matching module results, we used Precision@k and NDCG measures to measure how relevant the results are and how good the ordering is. Precision@k measures the proportion of relevant documents in the first K positions but disregards the individual position of each document. On the other hand, the Normalized Discounted Cumulative Gain (NDCG) as a measure of ranking quality is used to measure the order correctness of the selected ranked job list.

The NDCG@k metric involves finding the normalized discounted cumulative gain. The gain is just the relevance score for each item recommended, and cumulative gain at K is the sum of gains of the first K items recommended. Moreover, discounted cumulative gain weighs each relevance score based on its position. The recommendations at the top get a higher weight, while the relevance of those at the bottom gets a lower weight.

$$DCG@k = \sum_{i=1}^{K} \frac{G_i}{\log_2(i+1)} \tag{11}$$

The normalized discounted cumulative gain is the DCG with a normalization factor in the denominator. The denominator is the ideal DCG score when we recommend the most relevant items first.

$$NDCG@K = \frac{DCG@k}{IDCG@K} \tag{12}$$

Using these two measures, we compare the performance of the semantic matching module with three different retrieval algorithms: the Kullback-Leibler divergence algorithm (KL) (Manning, 2008), the Term Frequency-Inverse Document Frequency (TF-IDF; Zhai, 2008), and Okapi BM25 (BM25; Robertson, 1995). The KL divergence provides a non-symmetric measure of the difference between two documents. The TF-IDF is a numerical statistic that measures a word's importance to a document within a collection. It can be used as a model for comparing similarity between documents. Finally, Okapi BM25 is a bag-of-words retrieval model that ranks documents based on similar query terms in each document.

In our experiments, we randomly selected 10 candidate resumes; for each resume, we queried the corpus of 1000 jobs using the AnnoJob system to return the top 20 positions. Table 4 presents the quality of the recommendation results obtained using the proposed semantic matching approach compared with retrieval algorithms using Precision@k and NDCG. The semantic matching based on annotated documents and domain ontology and wikidata yields better results than the other retrieval algorithms. Furthermore, the AnnoJob system outperforms the existing information retrieval algorithms in the quality of matching the returned jobs.

Table 4. *Comparison of AnnoJob and retrieval algorithms results using NDCG and Precision@k.*

K	**AnnoJob**		**BM25**		**TF-IDF**		**KL**	
	NDCG	**P@k**	**NDCG**	**P@k**	**NDCG**	**P@k**	**NDCG**	**P@k**
5	0.8991	0.8249	0.1863	0.3002	0.4356	0.7810	0.3984	0.4230
10	0.7864	0.8001	0.1989	0.2133	0.4562	0.5630	0.4503	0.3789
20	0.7632	0.7893	0.2001	0.2001	0.4578	0.3817	0.4001	0.3842

Figures 18 and 19 show the results of comparing the AnnoJob and retrieval algorithms in recommending the best job for a given candidate. It is clear that AnnoJob produces the most consistent results compared to other techniques based on NDCG and Precision@k measures. Furthermore, the AnnoJob returns more relevant and stable Precision@k values than the other algorithms, especially the TF-IDF, which gives an accurate value in selecting the top 5 jobs and decreases when the selection area increases.

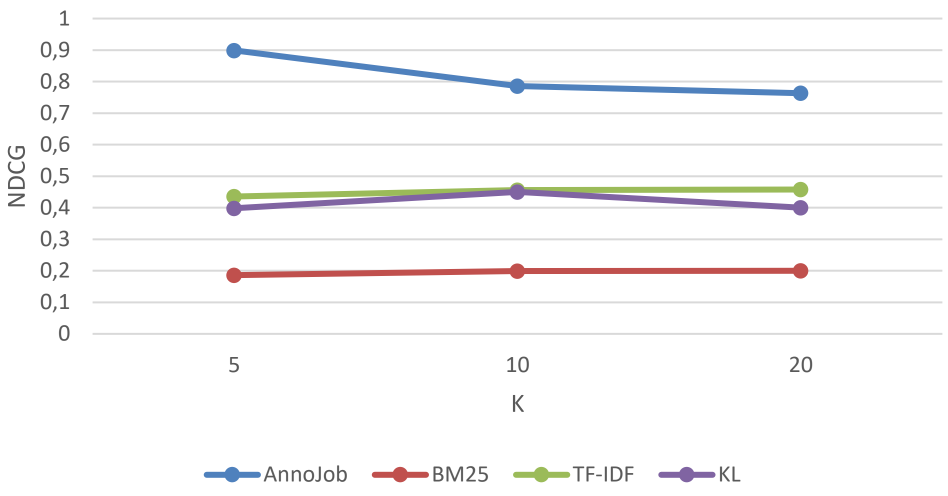


Figure 18. *Comparison of AnnoJob and retrieval algorithms results using NDCG.*

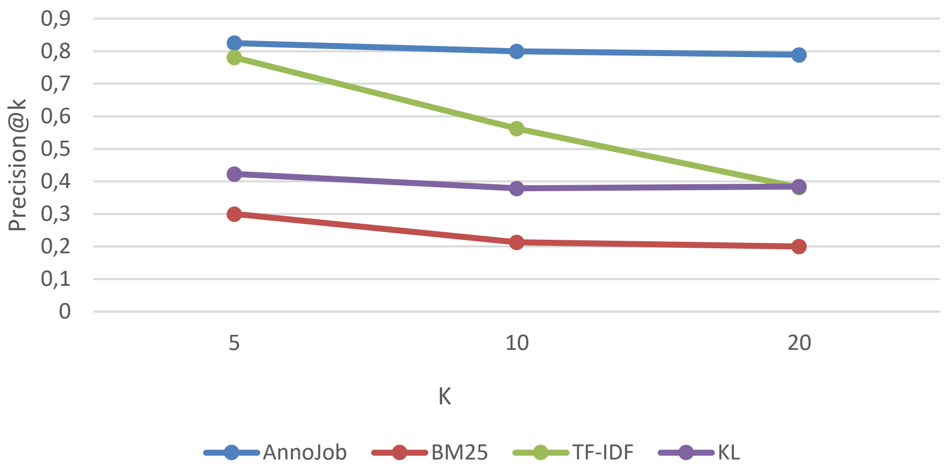


Figure 19. *Comparison of AnnoJob and retrieval algorithms results using Precision@k.*

5 Conclusions and Future Work

In this paper, we presented AnnoJob, a semantic annotation-based system. The proposed approach exploits the document' structure and lexical and syntactic rules to extract entities intelligently. We developed a domain ontology to interlink those entities with hierarchical and associative relations building a context. Finally, the extracted information (entities and links) are presented in RDF triples delivering a unified format for resumes and offers.

To match the job offers and resumes, we calculate the semantic distance between the documents, exploiting the RDF file structure and semantic resources (ontology and Wikidata) as a guide to find the semantic relatedness and similarity between entities. As a result, the system output is a ranked list of job offers relevant to a given resume. Moreover, the resume RDF file is enriched with the most relevant job offer titles, which will be the key to future job searches.

The assessment was performed on a data set of 1000 jobs and 100 resumes. The initial assessment was conducted by comparing verified data and system-extracted entities. We used the precision, recall and F-1 metrics to evaluate the information extraction module based on the semantic annotation approach using a domain ontology and JAPE rules.

The annotation quality and efficacy can be assessed by evaluating the results of the application that uses it. Therefore, we evaluated the matching module outcomes that exploit the annotated documents to rank a Top_K relevant job list for a given resume. We employed the Precision@K and NDCG measures to analyse the matching results, where the AnnoJob system gets better results than other information retrieval methods (KL, TF-IDF and BM25).

In our work, we presented the importance of:

- Extracting contextual entities and the semantic presentation of document information to improve job recommendation.
- Exploiting semantic relatedness and similarity using domain ontology and KG to match resumes and offers.

However, our proposal has limitations too, which are:

- Semantic-based methods give more accurate results than other approaches, such as machine learning-based approaches that require massive data for training and testing. However, the running time is the first obstacle for the semantic-based approach since the access, use and search over semantic resources take more time, especially using KG Wikidata from a distance via query service, where the query response depends on the network performance.
- The system cares about requirements but not about their importance level. For example, some requirements are considered must-have, important to have, or nice to have. However, the level can define the requirement priorities and therefore change the matching process by considering the weight of each requirement.

In the future, we plan to improve our system by:

- Adding a new component that intelligently browses job websites and collects offers to create an index of the latest job posts.
- Using fuzzy logic to determine the degree of relatedness between entities, which can enhance the precision of our system.
- Combining our approach with a machine learning method such as the BERT framework (Devlin et al., 2018) to improve the information extraction and the matching process time. The BERT framework can improve the pre-annotation phase and can be combined with the proposed semantic matching method to improve the results of the recommendations, since this technology proved its efficacity when it was combined with a semantic similarity measure such as Cosine or TF-IDF (Lavi et al., 2021).
- Considering the requirement priority in the matching process, since some requirements are strict and crucial.

Additional Information and Declarations

Conflict of Interests: The authors declare no conflict of interest.

Author Contributions: A.B.: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. Z.B.: Supervision, Writing – review & editing.

Data Availability: The data that support the findings of this study are available from the corresponding author.

References

Ahmed Awan, M. N., Khan, S., Latif, K., & Khattak, A. M. (2019). A New Approach to Information Extraction in User-Centric E-Recruitment Systems. *Applied Sciences*, 9(14), Article no. 2852. https://doi.org/10.3390/app9142852

Balachander, Y., & Moh, T. S. (2018). Ontology based similarity for information technology skills. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 302–305). IEEE. https://doi.org/10.1109/ASONAM.2018.8508726

Bizer, C., Heese, R., Mochol, M., Oldakowski, R., Tolksdorf, R., & Eckstein, R. (2005). The impact of semantic web technologies on job recruitment processes. In *Wirtschaftsinformatik 2005* (pp. 1367–1381). Physica. https://doi.org/10.1007/3-7908-1624-8_72

Bobadilla, J., Ortega, F. B., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge Based Systems*, 46, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Brek, A., & Boufaida, Z. (2020). Semantic Approaches Survey for Job Recommender Systems. In *RIF'20: The 91th Seminary of Computer Science Research at Feminine*. CEUR Workshop Proceedings. https://ceur-ws.org/Vol-3176/paper8.pdf

Brek, A., & Boufaida, Z. (2022). Enhancing Information Extraction Process in Job Recommendation using Semantic Technology. *International Journal of Performability Engineering*, 18(5), 369–379. https://doi.org/10.23940/ijpe.22.05.p7.369379

Çelik, D., & Elçi, A. (2012). An ontology-based information extraction approach for résumés. In *Joint international conference on pervasive computing and the networked world* (pp. 165-179). Springer. https://doi.org/10.1007/978-3-642-37015-1_14

Chen, J., Zhang, J., & Niu, Z. (2018). A Two-Step Resume Information Extraction Algorithm. *Mathematical Problems in Engineering*, 2018, 1–8. https://doi.org/10.1155/2018/5761287

Ciravegna, F., & Lavelli, A. (2004). LearningPinocchio: adaptive information extraction for real world applications. *Natural Language Engineering*, 10(2), 145–165. https://doi.org/10.1017/s135132490400333x

Daramola, J. O., Oladipupo, O. O., & Musa, A. G. (2010). A fuzzy expert system (FES) tool for online personnel recruitments. *International Journal of Business Information Systems*, 6(4), 444–462.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. https://arxiv.org/abs/1810.04805

Dhameliya, J., & Desai, N. (2019). Job recommender systems: A survey. In *2019 innovations in power and advanced computing technologies (i-PACT)* (Vol. 1, pp. 1-5). IEEE. https://doi.org/10.1109/i-PACT44901.2019.8960231

Färber, M., Bartscherer, F., Menne, C., & Rettinger, A. (2017). Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*, 9(1), 77–129. https://doi.org/10.3233/sw-170275

Färber, M., & Rettinger, A. (2018). Which Knowledge Graph Is Best for Me?. *arXiv preprint arXiv:1809.11099*. https://doi.org/10.48550/arXiv.1809.11099

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

Han, J., Zheng, L., Xu, Y., Zhang, B., Zhuang, F., Philip, S. Y., & Zuo, W. (2019). Adaptive deep modeling of users and items using side information for recommendation. *IEEE transactions on neural networks and learning systems*, 31(3), 737–748. https://doi.org/10.1109/TNNLS.2019.2909432

Heggo, I. A., & Abdelbaki, N. (2018). Hybrid information filtering engine for personalized job recommender system. In *International Conference on Advanced Machine Learning Technologies and Applications* (pp. 553–563). Springer. https://doi.org/10.1007/978-3-319-74690-6_54

Karaa, W. B. A., & Mhimdi, N. (2011). Using ontology for resume annotation. *International Journal of Metadata, Semantics and Ontologies*, 6(3/4), 166–174. https://doi.org/10.1504/ijmso.2011.048018

Kumaran, V. S., & Sankar, A. (2013). Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping (EXPERT). *International Journal of Metadata, Semantics and Ontologies*, 8(1), 56. https://doi.org/10.1504/ijmso.2013.054184

Lavi, D., Medentsiy, V., & Graus, D. (2021). consultantbert: Fine-tuned siamese sentence-bert for matching jobs and job seekers. *arXiv preprint arXiv:2109.06501*. https://doi.org/10.48550/arXiv.2109.06501

Fellbaum, C., & Miller, G. (1998). Combining local context and WordNet similarity for word sense identification. In *WordNet: An electronic lexical database* (265–283). MIT Press.

Lu, Y., El Helou, S., & Gillet, D. (2013). A recommender system for job seeking and recruiting website. In *Proceedings of the 22nd International Conference on World Wide Web* (pp. 963–966). ACM. https://doi.org/10.1145/2487788.2488092

Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing.

Maree, M., Kmail, A. B., & Belkhatir, M. (2019). Analysis and shortcomings of e-recruitment systems: Towards a semantics-based approach addressing knowledge incompleteness and limited domain coverage. *Journal of Information Science*, 45(6), 713–735. https://doi.org/10.1177/0165551518811449

Martinez-Gil, J., Paoletti, A. L., & Schewe, K. D. (2016). A smart approach for matching, learning and querying information from the human resources domain. In *East European Conference on Advances in Databases and Information Systems* (pp. 157–167). Springer. https://doi.org/10.1007/978-3-319-44066-8_17

Martínez-Rodríguez, J., Hogan, A., & Lopez-Arevalo, I. (2020). Information extraction meets the Semantic Web: A survey. *Semantic Web*, 11(2), 255–335. https://doi.org/10.3233/sw-180333

Mishra, R., & Rathi, S. (2021). Enhanced DSSM (deep semantic structure modelling) technique for job recommendation. *Journal of King Saud University – Computer and Information Sciences*, 34(9), 7790–7802. https://doi.org/10.1016/j.jksuci.2021.07.018

Mochol, M., Wache, H., & Nixon, L. (2007). Improving the accuracy of job search with semantic techniques. In *International Conference on Business Information Systems* (pp. 301–313). Springer. https://doi.org/10.1007/978-3-540-72035-5_23

Rácz, G., Sali, A., & Schewe, K. D. (2016). Semantic matching strategies for job recruitment: A comparison of new and known approaches. In *Foundations of Information and Knowledge Systems* (pp. 149–168). Springer. https://doi.org/10.1007/978-3-319-30024-5_9

Rácz, G., Sali, A., & Schewe, K. D. (2018). Refining semantic matching for job recruitment: An application of formal concept analysis. In *International Symposium on Foundations of Information and Knowledge Systems* (pp. 322–339). Springer. https://doi.org/10.1007/978-3-319-90050-6_18

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: introduction and challenges. In *Recommender systems handbook* (pp. 1–34). Springer. https://doi.org/10.1007/978-1-4899-7637-6_1

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/okapi_trec3.pdf

Al-Otaibi, S., & Ykhlef, M. (2012). A survey of job recommender systems. *International Journal of Physical Sciences*, 7(29). https://doi.org/10.5897/ijps12.482

Tran, M. L., Nguyen, A. T., Nguyen, Q. D., & Huynh, T. (2017). A comparison study for job recommendation. In *2017 International Conference on Information and Communications (ICIC)* (pp. 199–204). IEEE. https://doi.org/10.1109/INFOC.2017.8001667

Upadhyay, C., Abu-Rasheed, H., Weber, C., & Fathi, M. (2021). Explainable Job-Posting Recommendations Using Knowledge Graphs and Named Entity Recognition. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 3291–3296). IEEE. https://doi.org/10.1109/SMC52423.2021.9658757

Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. *Knowledge Based Systems*, 157, 1–9. https://doi.org/10.1016/j.knosys.2018.05.001

Yahiaoui, L., Boufaïda, Z., & Prié, Y. (2006). Semantic Annotation of Documents Applied to E-Recruitment. In *Proceedings of the 3rd Italian Semantic Web Workshop*. CEUR Workshop Proceedings. https://ceur-ws.org/Vol-201/10.pdf

Yi, X., Allan, J., & Croft, W. B. (2007). Matching resumes and jobs based on relevance models. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 809–810). ACM. https://doi.org/10.1145/1277741.1277920

Zhai, C. (2008). *Statistical language models for information retrieval*. Springer. https://doi.org/10.1007/978-3-031-02130-5

Editorial record: The article has been peer-reviewed. First submission received on 15 August 2022. Revisions received on 2 November 2022 and 27 December 2022. Accepted for publication on 30 December 2022. The editor in charge of coordinating the peer-review of this manuscript and approving it for publication was Stanislav Vojir.

Acta Informatica Pragensia is published by Prague University of Economics and Business, Czech Republic.

ISSN: 1805-4951