A Semantic Web enabled System for Résumé Composition and Publication

Roberto Mirizzi, Tommaso Di Noia, Eugenio Di Sciascio
Politecnico di Bari – Italy
mirizzi@deemail.poliba.it, {t.dinoia, disciascio}@poliba.it

Michelantonio Trizio
Data Over Ontological Models s.r.l. – Italy
michelantonio.trizio@doom-srl.it

Abstract

The process of writing a résumé is a task where the notion of background knowledge plays an important role. A typical résumé contains several interrelated and implicit information. The use of domain ontologies and semantic technologies provides a valuable help to point out such interrelations and to make explicit the implicit knowledge. We present a system to automatically produce a semantically annotated résumé, exploiting domain knowledge modeled with respect to a domain ontology. Semantic technologies and domain ontologies have been exploited both to help the user during the writing process and to explicitly represent domain knowledge in the final CV. Our aim is twofold: on the one hand a user obtains a document that can be stored in specific semantic-based systems used by recruiting agencies, and exposed on the web where, thanks to RDFa annotations can be indexed by emerging semantic-search engines.

1 Introduction

Writing a Curriculum Vitae (CV or résumé) is a fundamental task in everyone's life when approaching the job market. The more expressive and complete a CV is, the higher is the chance to be reached by a company.

The structure of a résumé is usually split into sections regarding: personal information; description of prior work experiences; description of education and training; detailed inventory of skills and competences acquired in the course of training, work and day life. The compilation of a résumé may follow two different structures: reverse chronological order and functional[1]. In the former, one presents positions and experiences listed in reverse order *i.e.*, starting with the most recent. Currently, this is the most used method. The latter represents experiences and skills sorted by area or job function. This type of writing allows the reader to focus on specific skills [6].

[1] Actually, there is also a hybrid form which balances the functional and chronological structure.

When the candidate starts typing her résumé she has to decide a method to adopt. Once this decision has been taken, it could be very difficult to turn back to the other one. However, depending on the application, reverse chronological order structure may be preferable over the functional one or vice versa. For instance, a functional résumé is preferable for applications to positions requiring a very specific skill set or clearly defined personality traits. On the one hand, this format highlights specific professional abilities and communicates the professional competency by a summary of the experiences. On the other hand, the chronological format emphasizes candidate's experiences by presenting a timeline of her career growth.

In order to produce a detailed and complete CV, before starting to write it down, the skilled worker should have in mind a clear picture of her skills, competences, knowledge. Although this could seem a trivial observation, actually this is one of the main difficulties when writing a résumé. The individual often does not detail all her previous experiences. Sometimes one might forget to mention particular skills, sometimes one might think they are not relevant or that they are implicit, and there is no need to make them explicit.

To illustrate such difficulties, let us consider an individual stating in her CV that she knows well `Oracle 10`[2] and not eliciting she knows `PLSQL`[3]. A traditional keyword-based retrieval system could not find a match between an employer seeking for a "`skilled PLSQL programmer`" even though `PLSQL` is usually implied by the knowledge of `Oracle 10`. The same is for a position where a "`programmer with experience on the`

[2] `http://www.oracle.com/database/index.html`
[3] `http://www.oracle.com/technology/tech/pl_sql/index.html`

Google Web Toolkit (GWT) framework"[4] is required. GWT is a Java-based framework but this is not explicitly defined in the position description. Hence, a résumé where there is only the keyword Java will not be considered at all as matching the job post. The examples make self-evident how much a user should be careful while writing her CV in order to avoid these situations.

In this paper we show how the use of domain ontologies can help in the composition and annotation of a résumé. In particular, we present a system that focuses on the process of writing and maintaining a CV guided by a semantic-aided user interface. Thank to a domain-ontology, the system is able to maintain and manage all the relations among skills, competences, knowledge and it may suggest implicit or related information to the user during the composition phase in the shape of a tag cloud. The outcomes of such a process are twofold. The user is guided through the composition and, exploiting ontology-based suggested information, she is helped in eliciting and making explicit her skills and competences. Moreover, the final résumé is annotated (tagged) w.r.t. a reference common ontological vocabulary. This means that a further search process of job candidates by employers (and dually of job offers by candidates) can be performed based on a semantic-based process.

Noteworthy is also that the final CV can be represented either using a chronological schema or a functional one with no effort from the user point of view.

The remainder of the paper is structured as follows. Next section is devoted to the description of semantic issues related to the composition of a CV. Then Sections 3.1, 3.2 and 3.3 show how to exploit ontological information to semantically enrich a résumé. Section 4 describes how to automatically generate a semantic-enabled CV using either a chronological approach or a functional one. Conclusions and description of future work close the paper.

2 A Semantic-enabled System for Resume Writing

Recent years have witnessed a proliferation of systems on the World Wide Web offering the ability for a job seeker to apply for a position, and on the opposite side for a company to seek for human resources. Such systems allow a user to insert her personal relevant information and this task is usually carried out by filling a sequence of textual forms. These forms are always divided into sections, usually according to a fixed chronological structure of a résumé. There is an initial section of Personal Information, including name, address, age, date of birth. This part is quite standard, with well structured information. The next step with the composition of the CV is the insertion of work experiences, usually starting with the most recent, according to the reverse chronological order previously mentioned. For each work experience the user specifies a starting date (From) and a final date (To). These two pieces of information are very important because they tend to represent the "level" of experience in the given field. Even in this step there are many information that remain hidden with respect to the explicit information introduced by the user. Each work experience - implicitly - implies the knowledge needed to do the work itself. A section about education and training usually follows after the section on work experiences. The structure of this part is very similar to the one described above and it suffers from the same problems. After the user has described her work experiences, she is asked to introduce her personal skills and competences. This section of a résumé includes social skills and competences, knowledge of languages, organizational competences, artistic skills, technical and computer competences. Especially in the world of ICT, a very important subsection is the one related to technical and computer skills. This section very often is examined by the recruiter to get an initial idea of the candidate. This section is very schematic and represents key competences and knowledge of the employee. Note that many informations have already been declared by the user when inserting her work experiences, but they could not be explicitly reported in the subsection related to technical and computer skills. For this reason the user has to pay attention to make evident here what she already mentioned before. Currently, systems do not create any link between the two sections, forcing the user to do that manually.

Moreover, each time a user updates her résumé, she has to insert the new experiences and add values to the list of competences, or modify existing ones by updating the dates. This means, for each new experience, she has to control two sections. Exploiting both ontologies and a tagging system, we will show how the user can reduce her effort updating just one part, with the system providing the completion of the other one.

The lack of explicit semantics in a résumé may result in a serious drawback during the retrieval phase. Among available systems we mention the CV parser developed by Sovren [9] or Faviki [8], devoted to extracting and structuring semantic information from a semi-structured document as a résumé is. The idea is that once the information is extracted then it is possible to semantically enrich and tag it with semantic-based annotation. Other systems embed semantic information within a résumé during the writing process (see hResume [3]). We combined these approaches and developed a system that using semantically enriched XHTML forms, via RDFa[5], and semantic-based tags produces a semantic-enabled version of a résumé[6].

[4]http://code.google.com/intl/it-IT/webtoolkit/

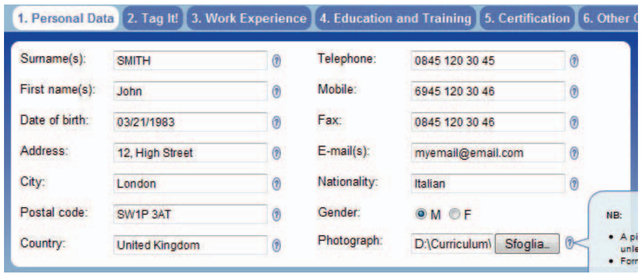
[5]http://www.w3.org/TR/xhtml-rdfa-primer/

[6]The system is available at http://sisinflab.poliba.it/

3 Proposed System

3.1 Personal Data

The XHTML form pictured in Figure 1 collects typical information presented in the *Personal Data* section of a résumé. Figure 2 shows the source of such page and how each input field is enriched with RDFa information. Information for each element of the form is tagged with the corresponding RDFa annotation in the XHTML file representing the final CV produced by the system.



Surname(s):	SMITH	Telephone:	0845 120 30 45
First name(s):	John	Mobile:	6945 120 30 46
Date of birth:	03/21/1983	Fax:	0845 120 30 46
Address:	12, High Street	E-mail(s):	myemail@email.com
City:	London	Nationality:	Italian
Postal code:	SW1P 3AT	Gender:	[✓ M] F
Country:	United Kingdom	Photograph:	D:\Curriculum\

Figure 1. Personal Data form

```
<div id="personal-data" about="#me"
   typeof="foaf:Person vcard:Name google:Person"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:vcard="http://www.w3.org/2006/vcard/ns#"
      xmlsn:oc="http://s.opencalais.com/1/pred/"
      xmlns:google="http://rdf.data-vocabulary.org/rdf.xml#"
      xml:lang="en-en">
    <div>
        <label for="name">Surname(s): </label>
        <input type="text" id="name"
           property="vcard:family-name foaf:family_name" />
    </div>

    <div>
        <label for="firstname">First name(s): </label>
        <input type="text" id="firstname"
           property="vcard:given-name foaf:givenname" />
    </div>
    [...]
    <div rel="vcard:homeAdr">
        <div typeof="vcard:Address google:PostalAddress">
            <div>
                <label for="address">Address: </label>
                <input type="text" id="address"
                   property="vcard:street-address
                             google:address" />
            </div>

            <div>
                <label for="city">City: </label>
                <input type="text" id="city"
                   property="vcard:locality
                             google:locality" />
            </div>
            [...]
        </div>
    </div>
    <div>
        <label for="nationality">Nationality: </label>
        <input type="text" id="nationality"
           property="oc:nationality" />
    </div>
    [...]
</div>
```

Figure 2. RDFa enhanced XHTML code for Personal Data form.

impakt-reloaded/.

Following the original idea of a Semantic Web [4], we semantically describe each input field of our XHTML forms to allow for an automated filling by a software agent. In fact, as a side effect, having a form semantically annotated can allow a software agent to "understand" the information needed to automatically fill the form. Moreover, annotating form fields becomes useful also to implement a content-based autocompletion suggestion (see Section 3.3).

3.2 Content-based and Collaborative filtering Tag Recommendation

In our system, we added a new initial layer to the process of résumé writing. This is a semantic-driven tagging step. The idea behind the need of this further step is to overcome the limits of existing web-based systems for résumé writing and to make more accurate the search phase. Currently, semantic tagging systems as `Faviki` suggest possible tags, exploiting a keyword-based search within a text. Automatically suggested tags do not cover all the knowledge modeled by the user while writing down her résumé. This could be a serious drawback as recruitment is a knowledge intensive task. The more accurate tags are, the more accurate search will result. We can summarize the search process in the following two simple steps:

(a) look for specific competences and skills. This is the information usually represented in job postings as **job requirements**: *"... eight or more years of direct experience in software development, including several years of hands-on development experience preferably in a complex distributed J2EE environment. Three or more years of experience as a hands-on manager of software development professionals. Experience in the health insurance industry and knowledge of Medicare concepts a strong plus. [...] Two to three years of project leader/technical lead experience required. Hands-on technical experience with distributed/web based J2EE/Oracle/Websphere systems is preferred. [...] Project Management skills and experience are desired. Excellent verbal and written communication skills required."* [7]

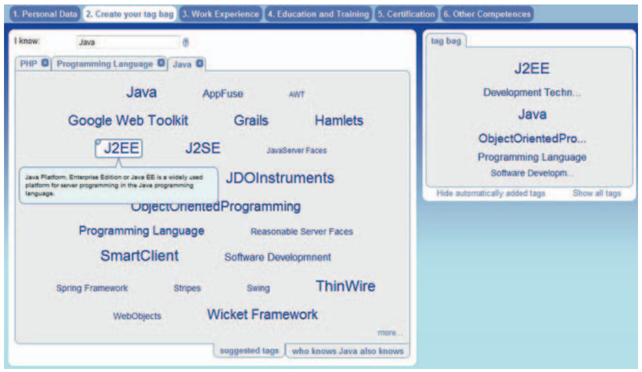
(b) match competence and skills with work experience for a second round retrieval step. This is the phase where textual information matter. The recruiter reads the description of work experience and manually extracts implicit and explicit information related to acquired competences and skills.

In the above described process, semantics plays a key role especially in the crucial first step. Indeed, how to retrieve

[7]http://www.careerbuilder.com/JobSeeker/Jobs/JobDetails.aspx?IPath=QHKTCV&ff=21&APath=2.21.0.0.0&job_did=J3I2GV6NKQ59S6HVJQW

résumés where the candidate declares to have *"eight or more years of direct experience in software development"*? The candidate usually just describes her work experience. She does not explicitly aggregate such information. But in this very common case we would need to recognize all the *"software development"* experiences of the candidate and put them all together to entail the *"years of direct experiences"*. Hence, linking work experiences with skills becomes a manually preformed task not only during the writing of a résumé (see Section 1), but also during the search phase. Here we propose a semantic-aided procedure to ease this linking process when writing a résumé. The idea is to start asking the user to elicit her personal competences and skills (and represent them as tags) and then to use this information to annotate the following textual descriptions in both the work experiences section and the education and training one. In other words, before starting to write down plain text about her work experiences, the user is asked to collect her skills and competences as a set of tags. Based on typed keywords, the system recommend [2] a set of possible tags which are semantically related to such keywords. The system uses an underlying OWL DL ontology where the domain of skills and competences is modeled and described. The ontology[8] mainly focuses on ICT domain and contains 3341 named `rdfs:class` [10] representing both so called hard skills and soft skills. The former refer to the background knowledge of a candidate on specific technologies and tools, the latter represent personal and social characteristics of the individual.

When the user starts typing a text, *e.g.*, "`jav`", the system suggests a list of possible tags that match with the characters written up to this moment, through an ajaxified autocompletion box. The suggested list represents the set of `rdfs:class` in the underlying ontology where the string `jav` appears either in the corresponding `rdfs:label` property or in the `rdfs:comment` one.
Based on the selected tag (the corresponding `rdfs:class`), for instance `Java` in our obvious example, the system suggests to the user other related tags represented in a classical tag cloud. The size of the single tag is computed according to its popularity with respect to all CVs previously stored within the knowledge base. In our running example the system would for example suggest `object oriented programming` and `JSP`. This a typical **content-based recommendation** procedure [2] where the content is represented by the underlying ontology. We can distinguish between two types of recommended tags: one that refers to something more general than `Java`, the other related to a more specific concept (see Figure 3). In other words, suggested tags represent RDFS classes which are either subclass or superclass of the one selected by the user. Suggested tags are sized according to specific criteria. We consider: (i) the distance of the selected class and the suggested one in the ontology[9]; (ii) the popularity of the suggested tag in other profiles stored within the knowledge base.



I know: Java

Java Platform, Enterprise Edition or Java EE is a widely used platform for server programming in the Java programming language.

Figure 3. Suggested Tag Cloud and user's Tag Bag

By looking at suggested tags a user can select the ones that better suit her skills. In our running example, the system would suggest concepts such as `JSP`, `J2EE` and other technologies/tools connected with `Java`. If the user has a specific competence, say `JSP`, she can select the tag from the cloud, with no need to type it explicitly or remembering to type it later. Moreover, if the user wants to see tags that are related to `JSP`, she can just click on the tag. A new tab will open with a new tag cloud. In such a way, the user is able to browse the ontology in a very natural and intuitive way. For each selected tag, the system shows two tabs with two different suggested tag clouds. The first one recommends content-based tags as described above, *i.e.*, it refers to the intensional part of the knowledge base (KB) managed by the system. The second one refers to the extensional knowledge of the KB, *i.e.*, to the CVs already stored within the knowledge base. In other words, in this tab we adopt a **collaborative-filtering** approach [5] for tags recommendation. As an example, given the selection of the tag `PHP`, the system suggests tags representing competences and skills owned by people who know `PHP`, looking at the annotations referring to CVs already stored within the knowledge base. In this case, it results that most PHP programmers have competence in `MySQL`, too. These two concepts are not strictly tight from an ontological/extensional point of view but they are highly coupled considering a common user experience. According to this coupling, the system could say that who knows `PHP` usually also knows `MySQL`

[8]The ontology is part of the skill management system I.M.P.A.K.T. developed by Data Over Ontological Models `http://www.doom-srl.it`

[9]In the current implementation, the distance is computed as the number of hops between two nodes in a tree.

with a certain degree of competence, `PostgreSQL` with a lower degree and `XHTML` with another different degree. Considering how common is a concept with respect to another is useful for sizing the tags in the cloud. In our example `MySQL` would have a larger size than `PostgreSQL`, while `XHTML` would have a size median with respect to the other two. Thanks to this tab, the user can quickly see how other profiles are composed and select a particular tag if it matches her competences or skills. Once a tag has been selected, it is added to the user's **Tag Bag** (see Figure 3).

In the context of writing a résumé, suggested tags may play a crucial role to help users in eliciting their own knowledge. In fact, it is a very common case finding CVs with few information not just because the candidate is not skilled or has a few competences. Very often the user does not state everything about her professional knowledge because she may consider some information not relevant, implicitly asserted or she simply completely forgets about them. From the point of view of an employer, exploiting the semantic nature of the tags, it would be possible to find particular competences also if they have not been explicitly added by the user. With respect to our example depicted in Fig. 3, an agency looking for a perspective candidate with `object oriented programming experience` could find the CV under examination suitable for its needs. Once the users has her Tag Bag populated by semantic tags, she can use them to semantically-annotate the rest of her résumé.

3.3 Semantic Tagging of a CV

After the above step, the user's Tag Bag has been populated by tags representing her global skills and competences. At this point she is ready to link such global information to a specific work experience. In the Work Experience section (see Figure 4) the user fills the corresponding form and she can describe her experiences. After filling the form, she can use tags previously collected in the Tag Bag to annotate the corresponding work experience. In addition to the advantage of producing a semantically-tagged document, a further benefit ensues. Going back to the example in Section 3.2, we can solve the issue of aggregating information and competences: "*[...] eight or more years of direct experience in software development [...]*". In fact, suppose the user tags with `Java` a work experience of six years and with `PHP` another one of three years. Then, since both Java and PHP relate to `Software Development`, we can aggregate the information related to both work experiences and entail a *nine years experience in software development*. Hence, the level of competence in a specific or generic field can be computed exploiting implicit information hidden within the semantics of the tags. The same procedure is used to compile the *Education and Training* section of the CV. Even in this section, semantics is crucial. As an example we just cite the case of certifications. A Cisco Certified Network Associate (CCNA) certification implies the knowledge of how to install, configure, operate, and troubleshoot enterprise level router and switched networks.

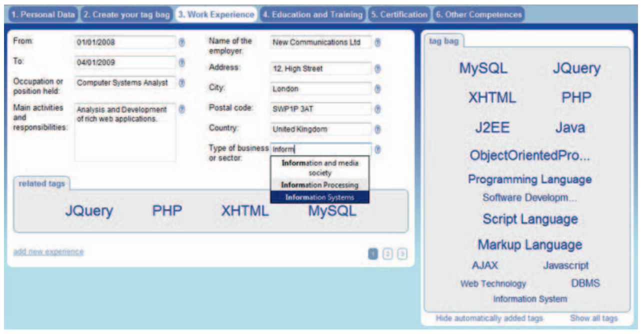
We point out that exploiting the semantic annotations associated to form fields, the systems is able to perform a semantic based autocompletion suggestion. For instance, the RDFa annotation:

```
<div
  xmlns:impakt="http://www.doom-srl.it/impakt/pred/">
    <label for="employer-type-of-business">
    Type of business or sector:
    </label>
    <input type="text" id="employer-type-of-business" property="impakt:hasIndustry" />
</div>
```

of the input field *Type of Business or Sector* in Figure 4 means that the content of such field has to be a business sector. In RDF words, since in the ontology we have:

```
<rdf:Property rdf:about="impakt:hasIndustry">
    <rdfs:label>has Industry</rdfs:label>
    <rdfs:comment>
       State the nature of the employer's business or sector
    </rdfs:comment>
    <rdfs:domain rdf:resource="doac:Experience"/>
    <rdfs:range rdf:resource="impakt:Industry"/>
</rdf:Property>
```

the suggestion list will be populated by items representing subclasses of `Industry` ruling out all those suggestions that match only with respect to a string comparison. Here, the autocompletion suggestion strongly relates to the content of the input field represented by its semantic annotation.



From:	01/01/2008	Name of the employer:	New Communications Ltd
To:	04/01/2009	Address:	12, High Street
Occupation or position held:	Computer Systems Analyst	City:	London
Main activities and responsibilities:	Analysis and Development of rich web applications.	Postal code:	SWP1P 3AT
		Country:	United Kingdom
		Type of business or sector:	Inform

Information and media society
Information Processing
Information Systems

Figure 4. Annotated work experience

4 Automated Generation of a Semantically Annotated CV

The final outcome of the system is a XHTML file containing all information provided by the user together with related semantic annotation. In other words, the final file is a fully RDFa compliant XHTML file representing the user's CV. Moreover, each section of the file is annotated and semantically described. Some snippets of the RDF annotation related to information described in Figure 4, is reported below. Note that since all information related to user's skills and experiences are semantically represented, then it is possible to represent such an information either from a functional point of view or using a chronological approach. In fact, in order to aggregate work experiences with respect to a particular function it is enough to refer to corresponding semantic annotations. As an example, in case the user is willing to aggregate all her experiences in "Information Systems" the system collects all the experiences whose annotation relates to `impakt:Information_Systems`.

```
<doac:experience>
  <doac:Experience rdf:about="base:workExperience">
   <impakt:hasIndustry
    rdf:resource="impakt:Information_Systems"/>
   <impakt:hasJobTitle
    rdf:resource="impakt:Computer_Systems_Analyst"/>
   [...]
   <doac:end-date>2009-04-01</doac:end-date>
   <doac:start-date>2008-04-01</doac:start-date>
   <impakt:hasKnowledge rdf:resource="impakt:Mysql"/>
   <impakt:hasKnowledge rdf:resource="impakt:jQuery"/>
   <impakt:hasKnowledge rdf:resource="impakt:XHTML"/>
   <impakt:hasKnowledge rdf:resource="impakt:PHP"/>
   <doac:activity>Analylsis and Development of web
         applications</doac:activity>
 </doac:Experience>
</doac:experience>
```

In the Appendix we show a snippet of the final XHTML output of the CV.

5 Conclusion and future work

In this paper we described a system fully exploiting semantic technologies to create a semantic-enabled résumé. Semantics is pervasive within the whole system. On the one side, each page has been enriched via RDFa annotations, starting form input fields in XHTML web forms. Thanks to RDFa, the field self-describes its expected content and allows a software agent to automatically fill the form. On the other side, a domain ontology has been exploited to help the user in creating her own tag bag, *i.e.*, a set of tags (representing classes of the underlying ontology). A user can exploit such (semantic-enabled) tags to annotate sections of the résumé. Finally, the system produces a XHTML file where tags and other semantic information are materialized as RDFa statements. Such pages can be exposed on the Web ready to be crawled *e.g.*, by semantic agents from Yahoo![1] and Google[7], just to cite a few. We are currently working on a new version of the ontology, exploiting standard ontology based vocabularies. In particular we are looking at how to extract information which is relevant for our domain from DBpedia[10]. As a parallel task, we are also investigating how to perform an efficient (from the final user perspective) retrieval of semantically annotated CVs exploiting the structure of DBpedia.

References

[1] Accessing Structured Data using BOSS. `http://developer.yahoo.com/search/boss/structureddata.html`, viewed May 16 2009, 2009.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[3] J. Allsopp. *Microformats - Empowering Your Markup for Web 2.0*, page 212. Springer, 2007.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, pages 230–237, 1999.

[6] Jobera Chronological Resumes. `http://www.jobera.com/job-resumes-cvs/resumes/chronological-resumes.htm`. viewed 23 April 2009, 2009.

[7] M. Mayer and J. Menzel. More search options and other updates from our searchology event. `http://googleblog.blogspot.com/2009/05/more-search-options-and-other-updates.html`, 2009.

[8] V. Milicic. Semantic web use cases and case studies – case study: Semantic tags. `http://www.w3.org/2001/sw/sweo/public/UseCases/Faviki/`, 2008.

[9] I. Sovren Group. Overview of Sovren's Proven Technology Offerings. `http://www.sovren.com/`, 2007.

[10] E. Tinelli, A. Cascone, M. Ruta, T. Di Noia, E. Di Sciascio, and F. M. Donini. I.M.P.A.K.T.: an innovative, semantic-based skill management system exploiting standard SQL. In *11th International Conference on Enterprise Information Systems (ICEIS'09)*, 2009.

[10] `http://dbpedia.org/`

Appendix

```
...
<table about="#me" typeof="foaf:Person vcard:Name google:Person">
  <tr><td>Work experience</td>
       <td><span rel="doac:experience">
       <span about="#workExperience" typeof="doac:Experience">
        <span rel="doac:organization"><span about="#organisation" />
       </span></span></span>
      <span about="#workExperience" property="doac:start-date"
           content="2008-04-01">01 April 2008</span> -
      <span about="#workExperience" property="doac:end-date"
           content="2009-04-01">01 April 2009</span></td></tr>
  <tr><td>Occupation or position held</td>
    <td about="#workExperience" property="google:role doac:position">Systems analyst</td>
  </tr>
  <tr><td>Main activities and responsibilities</td>
    <td><span  about="#workExperience" property="doac:activity">
      Analylsis and Development of web applications</span>
       <span property="impakt:hasKnowledge">jQuery</span>
       <span property="impakt:hasKnowledge">PHP</span>
       <span property="impakt:hasKnowledge">XHTML</span>
       <span property="impakt:hasKnowledge">MySQL</span>
    </td></tr>
  <tr><td>Name and address of employer</td>
    <td rel="ov:businessCard">
      <div typeof="vcard:VCard">
       <div rel="vcard:org">
        <div about="#organisation"
         typeof="vcard:Organization google:Organization foaf:Organisation">
         <div property="vcard:organization-name">New Communications Ltd</div>
         <div rel="vcard:workAdr">
          <div typeof="vcard:Address">
           <span property="vcard:street-address">
            12, High Street</span>,
           <span property="vcard:locality">London</span>
      </div></div></div></div></div>
  </td></tr>
    <tr><td>Type of business or sector</td>
    <td property="impakt:hasIndustry">Information Systems</td></tr>
...
```