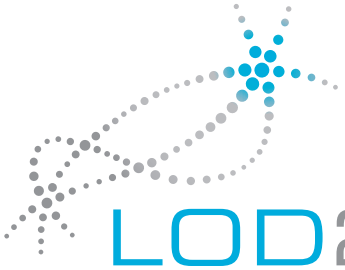

Collaborative Project

LOD2 - Creating Knowledge out of Interlinked Data

Project Number: 257943 **Start Date of Project:** 01/09/2010 **Duration:** 48 months

Deliverable 9a.2.2

Stable Implementation of Matching Functionality into Web Application for Filing Public Contracts

Dissemination Level	Public
Due Date of Deliverable	Month 40, 2013-12-31
Actual Submission Date	Month 48, 2014-08-20
Work Package	WP 9a, LOD2 for a Distributed Marketplace for Public Sector Contracts
Task	T9a.2
Type	Prototype
Approval Status	Approved
Version	1.0
Number of Pages	18
Filename	deliverable-9a.2.2.pdf

Abstract: The deliverable presents an updated version of the API for matchmaking web services, discusses evaluation of matchmaking quality and describes the integration of the matchmaker's API into the Public Contracts Filing Application. A preliminary experiment in annotation of public contracts with product ontologies is described and its potential contribution to matchmaking is examined.

Project funded by the European Commission within the Seventh Framework Programme (2007 - 2013)

History

Version	Date	Reason	Revised by
0.1	2014-03-11	Initial version	Jindřich Mynarz
0.2	2014-05-12	Description of an annotation experiment	Marek Dudáš
0.7	2014-07-05	API description and evaluation results	Jindřich Mynarz
0.8	2014-07-21	Updated section on integration	Jindřich Mynarz, Václav Zeman
0.9	2014-07-30	Review comments from Katja Eck (WKD) addressed	Jindřich Mynarz
1.0	2014-08-19	Review comments from Martin Kaltenböck (SWC) addressed	Jindřich Mynarz

Author List

Organization	Name	Contact Information
UEP	Jindřich Mynarz	jindrich.mynarz@vse.cz
UEP	Václav Zeman	vaclav.zeman@vse.cz
UEP	Marek Dudáš	marek.dudas@vse.cz

Executive Summary

The main changes in the design of matchmaker's API include transition from JSON partially mapped to RDF to JSON-LD responses that can be fully mapped to RDF data model. The API responses expressed with the Hydra Core Vocabulary[1] have self-describing representations with embedded hypermedia controls, which generic automated clients may use to navigate the API. The updated matchmaker API benefits from a better design that cleanly separates loading data and matchmaking.

The matchmaker API was integrated into the Public Contracts Filing Application using a proxy that is accessed via JavaScript calls from the application's user interface. A demo instance of the matchmaker API is publicly available at http://lod2.vse.cz:8080/matchmaker/ and an instance of Public Contracts Filing Application that uses the API can be found at http://lod2.vse.cz:8080/pc-filing-app/.

Evaluation of matchmaking performance indicates that a simple matchmaking method based on exact matches of Common Procurement Vocabulary codes outperforms other approaches both in relevance of results and in response time. The matchmaking algorithm is not the only factor in matchmaking quality. Preliminary experiments suggest that data quality in general, and deduplication of business entities in particular, has a large impact on matchmaking that is based on data aggregation.

[1] http://www.hydra-cg.com/spec/latest/core/

Table of Contents

1 Introduction

The deliverable presents a prototype implementation of web services matchmaking public contracts and business entities and the integration of these services into the user-facing Public Contracts Filing Application (PCFA). Results of the presented work are embodied in 2 software artefacts; an updated matchmaker API and PCFA which has the matchmaker API integrated. The integration of matchmaking functionality was done by interacting with the matchmaker's API through a proxy web service that enriches and forwards matchmaker's results to the user interface of PCFA.

Compared to the preceding deliverable on matchmaking web services [4], the progress described in following report comprises improved API exposed by the matchmaking services and an example integration of these services into a user interface of the Public Contracts Filing Application. The planned usability tests were postponed to the upcoming deliverable 9a.3.2 in this work package, so that both matchmaking and analytical functionality described in the following deliverable can be tested at once.

All vocabulary prefixes used in the following text can be resolved to their corresponding namespace URIs via `http://prefix.cc`.

2 Matchmaker's API

The main changes to the matchmaker's API compared to its previous version [4] comprise fully RDF data, self-describing API responses with hypermedia controls, and cleaner API design. While the previous matchmaker provided JSON responses partially mapped to RDF via JSON-LD context, the current matchmaker's API offers all data in RDF using the JSON-LD serialization [5]. Each part of the API, including errors, is described in JSON-LD and can be read and acted upon by machine clients. The API responses are described using Hydra Core Vocabulary [2], by which it aims to provide enough data to enable autonomous machine clients to navigate through the API.

Hydra offers means to describe web APIs in terms of resources and operations performed on them. The goal of Hydra is to provide API description that can be both read and interacted with by machine clients. Hydra APIs are meant to be self-describing, so that clients can discover API functionality during runtime, without relying on prior preconfigured knowledge about the API. In the ideal case, to which Hydra aspires, a well-described API should be usable to generic clients without any out-of-band information, including external documentation.

The matchmaker's API is built using the fundamental concepts of Hydra, including supported classes and operation. It also uses several concepts that are recognized as under-specified in Hydra and marked with testing status, such as templated links. Additionally, Hydra was combined with other vocabularies to provide richer representations of API resources (e.g., providing example values for IRI template mappings with `skos:example`). Some of these vocabulary terms cover uses that Hydra has been proposed to cover but does not cover them so far. Given that these features are not stable and their expected interpretation is not yet codified, their use within the matchmaker's API may change in the future.

A related vocabulary that was considered for adoption is Schema.org actions [1]. However, we chose Hydra over Schema.org actions because it offers a cleaner and more mature design. An example downside of the Schema.org actions vocabulary is that it relies on microsyntaxes embedded in identifiers (e.g., `*-input` and `*-output` properties), which are inaccessible to the RDF data model.

JSON-LD was chosen as the default syntax of the API's responses because it provides a connecting link between semantic web technologies and regular web technologies. The syntax may be processed with a JSON parser, which makes JSON-LD data more accessible to non-RDF data consumers. It can also simple to manipulate in code, because it can be converted to native programming language structures, such as hash maps.

2.1 API operations

The matchmaking API describes its functionality in terms of operations that clients may perform by sending HTTP request. These operations are available for instances of the classes supported by the API, which include public contracts (`pc:Contract`) and business entities (either `gr:BusinessEntity` or `schema:Organization`). The matchmaker supports 2 kinds of operations: load and match. The distinction between these operations provides for a cleaner separation of the matchmaker's functionality compared to its previous version.

2.1.1 Load operations

Load operations allow to apply matchmaking to external data. An API client may upload an RDF representation of any of the supported classes serialized either in JSON-LD or Turtle. Corresponding to supported classes,

there are 2 endpoint URLs for load operations: `/load/business-entity` and `/load/contract`. In order to be accepted, the loaded representation must describe exactly 1 instance of the loaded class. Otherwise it cannot be automatically determined what resource should be used for matchmaking. The load operations may be executed by issuing a HTTP PUT request to a load operation endpoint URL with a representation of the resource in the request's body. If the operation is successful, its response contains a temporary named graph URI assigned to the uploaded representation, which a client can pass to matchmaking operations via the `graph_uri` parameter. The response includes generated links to match operations that are available for the uploaded resource. The graph URI of the uploaded resource is temporary because there is a periodically executed job that deletes older uploaded data using a pre-configured interval. Load operations can also be dereferenced via HTTP GET method to retrieve their descriptions.

2.1.2 Match operations

Match operations exposed by the API can be executed by issuing HTTP GET request. The URI of the matched resource is provided to match operations as a GET parameter. This allows to reference their results by the URI only and also makes it possible to cache the results. Using the URI of the matched resource, the matchmaker can fetch the resource's representation from its RDF store, and thus obtain data that is used for matching, such as CPV codes. As was the case for the previous matchmaker described in [4], there are 3 HTTP endpoints corresponding to 3 combinations of the supported classes:[2]

1. **Match business entity to contracts**

   The operation matches a business entity to public contracts that might be of interest for the entity.

   Endpoint: `/match/business-entity/to/contract`

2. **Match contract to business entities**

   The operation matches a contract to business entities that may be interested in bidding for the contract.

   Endpoint: `/match/contract/to/business-entity`

3. **Match contract to contracts**

   The operation matches a contract to similar contracts.

   Endpoint: `/match/contract/to/contract`

Input

All match operations accept common parameters described in table 1. Match operations retrieving public contracts include additional parameters described in table 2. Each of the input parameters is described in a dereferenceable IRI template mapping (`hydra:IriTemplateMapping`), which documents how the parameter should be mapped to a property of input data, states if it is required, and provides its default and example value. For example, a request to match contract to suitable business entities may simply use the contracts's URI as a value of the `uri` query parameter: http://lod2.vse.cz:8080/matchmaker/match/contract/to/business-entity?uri=http%3A%2F%2Flinked.opendata.cz%2Fresource%2Fvestnikverejnychzakazek.cz%2Fpublic-contract%2F368433-7403021068433.

Parameter	Required	Default	Possible values	Description
uri	yes	-	URI	URI of a matched entity
matchmaker	no	exact-cpv	exact-cpv, expand-to-narrower-cpv	Identifier of matchmaker to be used
limit	no	10	positive integer	maximum number of results to return
offset	no	0	non-negative integer	0-indexed offset of the first result to return
graph_uri	no	taken from configuration	URI	URI of the matched resource's named graph

Table 1: Common parameters of matchmaking operations

Parameter	Required	Default	Possible values	Description
current	no	false	xsd:boolean	flag to limit results to current (not awarded) contracts
earliest_creation_date	no	-	xsd:date	the earliest date when a relevant contract could be created
publication_date_path	no	-	SPARQL 1.1 property path	SPARQL 1.1 property path to contract's publication date

Table 2: Additional parameters of operations matching to contracts

Output

Responses of match operations provide paged collections of results serialized in JSON-LD. The results are sorted in a descending order according to their score assigned by the matchmaker. Each result is described by its URI, its score assigned by the matchmaker, and a label if available. The amount of data describing results is intentionally kept minimal in order to avoid increased query and data transmission cost. What additional data about matchmaker's results is needed depends on specific use cases, which is why the API responses include only the data that is necessary for most use cases. If more data needs to be fetched, the URIs of results may be templated into a query that retrieves the data that is needed in the given case.

2.2 Implementation

The matchmaker offers 2 methods how to compute score of the matches. Both methods are based on aggregation of overlaps of Common Procurement Vocabulary (CPV)[3] codes between the matched resource and

[2]The remaining combination, matching business entities to similar business entities, is not implemented.
[3]http://simap.europa.eu/codes-and-nomenclatures/codes-cpv/codes-cpv_en.htm

```
{
  - @context: [
        "http://www.hydra-cg.com/spec/latest/core/core.jsonld",
        "http://lod2.vse.cz:8080/matchmaker/jsonld_contexts/matchmaker_api.jsonld"
    ],
  - member: [
      - {
            @type: "gr:BusinessEntity",
            @id: "http://linked.opendata.cz/resource/business-entity/a6a05b94-9900-4b17-bb70-067fb91d7234",
          - vrank:hasRank: {
                vrank:hasValue: 63
            },
            gr:legalName: "HEWLETT- PACKARD s.r.o."
        },
      - {
            @type: "gr:BusinessEntity",
            @id: "http://linked.opendata.cz/resource/business-entity/1b3a0edc-372a-4ecd-9775-3d29d78b689c",
          - vrank:hasRank: {
                vrank:hasValue: 45.16
            },
            gr:legalName: "ASD Software, s.r.o."
        },
      - {
            @type: "gr:BusinessEntity",
            @id: "http://linked.opendata.cz/resource/business-entity/5f3c7648-ad16-457c-94de-038197a148e1",
          - vrank:hasRank: {
                vrank:hasValue: 42.24
            },
            gr:legalName: "AQUASOFT, spol. s r.o."
        },
```


Figure 1: Example of matchmaker's results in JSON-LD

the resources to which it is matched. CPV codes are equipped with labels in all official languages of the EU, so they are not language-specific, which can help in bridging between EU member states in cross-country procurement. CPV codes describe the objects of contracts in the EU public procurement, which these methods use to approximate semantic similarity of the matched resources described with CPV. The methods can distinguish between main object (`pc:mainObject`) and additional objects (`pc:additionalObject`) by configuring a factor from the interval $[0, 1]$ that inhibits the score gained via additional objects. Both methods are implemented as SPARQL 1.1 queries.

The first one, identified as `exact-cpv`, aggregates exact CPV matches between the matched resources. Since it performs only exact matches via efficient joins, the method has a fast response time. The second method, identified as `expand-to-narrower-cpv`, extends the first method with transitive expansion of CPV codes to narrower codes using the CPV's hierarchical structure. Because of the higher number of joins this method needs to perform, it has a slower response time than the first one.

In general, the steps performed by SPARQL processor executing matchmaking queries can be summarized in the following way:

1. Select CPV codes of the matched resource
2. Join matching resources based on their associated CPV codes
3. Aggregate partial score for each CPV code of each matching resource
4. Aggregate scores of the CPV codes associated with each matching resource
5. Order by score and limit to top N results

6. Select labels for each matching resource
7. Aggregate labels to randomly select 1 label per matching resource

The matchmaker offers a basic temporal filtering if its results are public contracts. Using the `current` boolean parameter a client may indicate that the matchmaker's results must include only those contracts that are still current, i.e. their tender submission deadline is in the future. The `earliest_creation_date` parameter offers a way to limit the maximum age of contracts of interest. In this way, only the contracts published after the specified date will be included in the matchmaker's results. Having this filter is useful for recurring execution of matchmaking queries, such as for a subscription service that periodically retrieves new matches. Additional filtering may be provided by retrieving the matchmaker's results, fetching required additional data for each result, and selecting results that satisfy the filter criteria. Incorporation of more attributes, including unstructured or data-typed literal, into the matching process is a matter of future work.

Matchmaking web services were implemented using the Clojure programming language.[4] Since Clojure is hosted on the Java Virtual Machine, it enables access to the available Java libraries, so that it is possible to reuse existing implementations of semantic web technologies and standards, while working with a language that offers a better expressivity than Java. The services are built on SPARQL 1.1-compliant endpoint, such as OpenLink's Virtuoso 7[5] or Jena Fuseki.[6] The matchmaker's implementation is based on the SPARQL 1.1 recommendations, including SPARQL 1.1 Query[7] and Update,[8] and SPARQL Graph Store HTTP protocol.[9]

A demo instance of the matchmaker API is publicly available at http://lod2.vse.cz:8080/matchmaker/. Source code of the matchmaker can be found in a public repository[10] licensed as open source under the terms of Eclipse Public License.[11] Deployment instructions can be found in the source code repository.

[4] http://clojure.org/
[5] https://github.com/openlink/virtuoso-opensource/tree/develop/7
[6] http://jena.apache.org/documentation/serving_data/
[7] http://www.w3.org/TR/sparql11-query/
[8] http://www.w3.org/TR/sparql11-update/
[9] http://www.w3.org/TR/sparql11-http-rdf-update/
[10] https://github.com/opendatacz/matchmaker
[11] https://www.eclipse.org/legal/epl-v10.html

3 Evaluation

In order to enable comparison of matchmaking methods and their configurations we devised 3 metrics that indicate the performance of matchmaker configurations:

1. percentage of cases in which correct matches were found in top N matchmaker results
2. average rank of correct match in matchmaker's results
3. average response time

In all metrics the correct match is the match provided by a ground truth dataset.

3.1 Evaluation setup

The ground truth dataset, to which matchmaker's results are compared, consists of retrospective awarded contracts from the Czech public procurement register.[12] Preparation of the Czech procurement data is described in detail in [6, p. 18]. Given such basis, the evaluated task is the prediction of contract winners. In this setup, the correct match for a contract is its winning bidder. Viewed from this perspective, the matchmaker is trying to approach the quality of selection by contracting authorities. However, it is apparent that this approach is subject to limitations. The selection of contract's winner by contracting authority may not only reflect the suitability of the winner, but it can also be skewed by adverse factors including insufficient information, clientelism, or corruption. Matchmaking based on prior awarded contracts also suffers from several biases because it favours larger and older business entities over newcomers to the procument market.

An alternative ground truth may be obtained from domain experts who can annotate public contracts with business entities that they consider as suitable suppliers for a given contract. However, this approach may work only for contracts from a limited domain, in which a domain expert is likely to orient well. Evaluation can be also done by having domain experts to judge appropriateness of suggested matches for particular contracts. Similarly to the previous approach, this one also needs experts on the specific domains of the contracts considered. Moreover, this way of evaluation cannot be reused for more than one matchmaker's configuration. If the goal is to compare performance of multiple matchmaker configuration, their results need to be manually evaluated for each configuration.

Because of the limitations of these alternative approaches to evaluation of matchmaking quality, we opted for the first approach using retrospective ground truth despite its known biases. A measure that may improve this approach is to take into account for evaluation only the contracts issued by "well-behaving" contracting authorities. Following this reasoning, we are considering to use the database of zIndex,[13] which provides rating of transparency and quality of tenders for each Czech contracting authority.

Due to the choice of ground truth, the evaluation was done for the service matching contracts to business entities. The evaluation setup consisted of 10 runs, the results of which were averaged. Each run randomly selected a sample of 100 contracts, for which award data was withheld from the matchmaker, and for each contract retrieved first 100 matches provided by the matchmaker. The evaluation was perfomed on the updated Czech public procurement dataset [6, p. 18] consisting of 10.18 million triples containing 188 thousand instances of contracts (i.e. `pc:Contract`). The matchmaker was run on a server with the LOD2 Stack distribution, 16 GB RAM, and 4 CPU cores with 3.2 GHz.

[12] http://vestnikverejnychzakazek.cz/
[13] http://www.zindex.cz/en/

3.2 Results

The evaluation results suggest that the `exact-cpv` matchmaker performs the best in all 3 metrics. Because of its performance we selected it as the default matchmaking method.

The evaluation results are depicted in the figure 3.2, which shows a comparison of 2 matchmaking methods: exact CPV (`exact-cpv`) and expanded CPV (`expand-to-narrower-cpv`), and results of the exact CPV matching on richer data samples. In all compared cases, the data samples used for the evaluation were selected randomly from the subset of public contracts, for which the awarded tender is known. To illustrate the influence of data on the performance of matchmaking, a case using data samples restricted to contracts featuring richer description[14] was added.

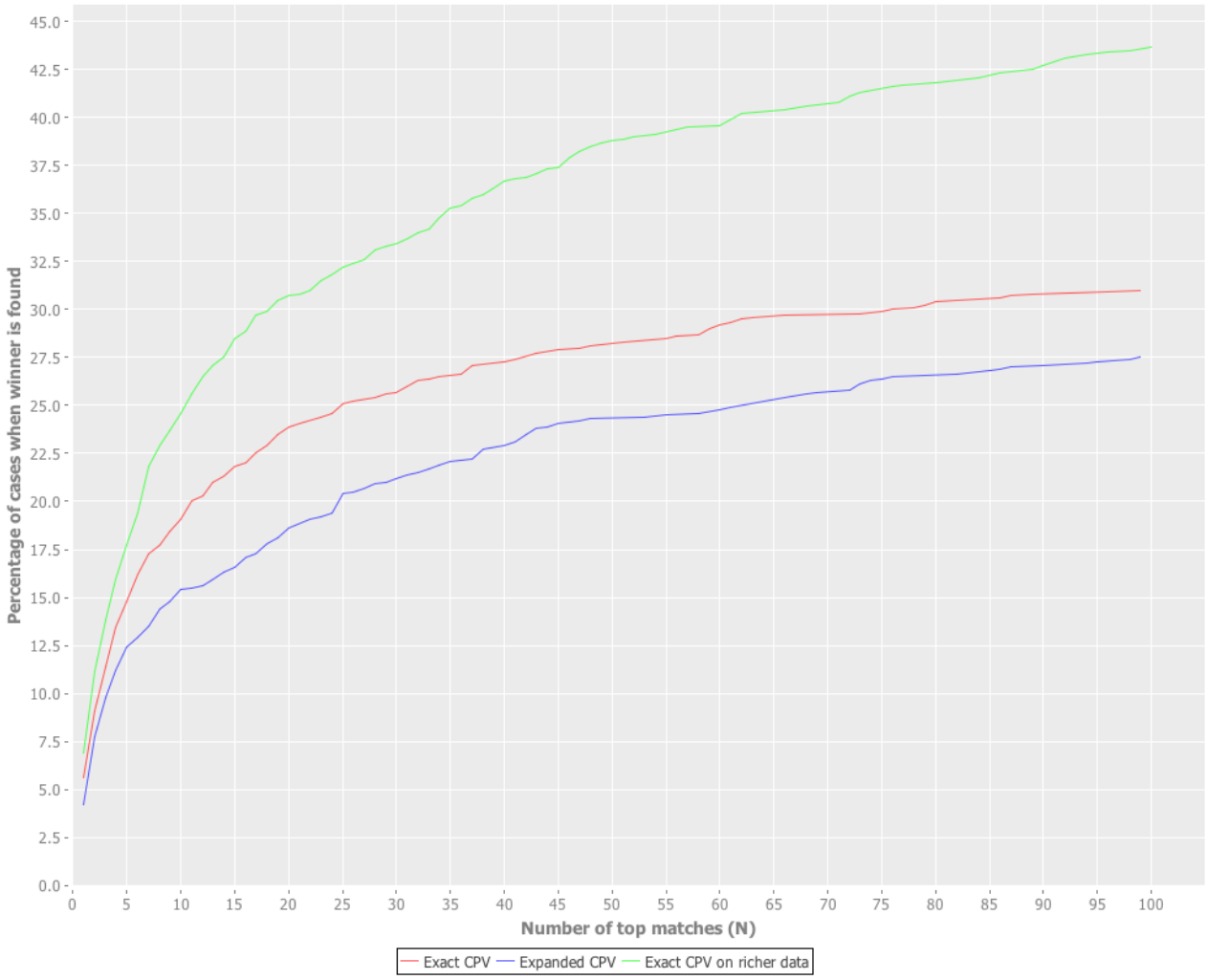

Figure 2: Ratio of winners found in top N matches

The figure represents the ratio of cases in which contract winners were found in top N matchmaker's results. The independent variable (N) is the number of matchmaker's results sorted in descending order that are taken into account. The dependent variable represents the percentage of cases in which the correct match is found in N matchmaker's results. The following table 3 summarizes evaluation results for the 3 discussed

[14] The samples were restricted to contracts having at least 1 CPV code as main object (`pc:mainObject`) and at least 3 CPV codes as additional objects (`pc:additionalObject`).

configurations. It highlights the correct matches found in top 10 matchmaker's results, which is likely the only part of results that is considered by users.[15]

Configuration	Correct matches found in top 10 results	Average rank of correct match in top 100 results
`exact-cpv`	19.1 %	15.43
`expand-to-narrower-cpv`	15.4 %	19.26
`exact-cpv` on richer samples	24.6 %	19.08

Table 3: Summary results of matchmaking configurations

The third chosen evaluation metric was the average response time of the tested matchmakers. In general, the current matchmaker exhibits a significant speedup over the previous implementation. It provides an average sub-second response time. For example, the `exact-cpv` matchmaker on Virtuoso 7.0 back-end responds in 0.14 seconds on average using the previously described setup. The response time of the matchmaker's predecessor ranged in tens of seconds.[16] This improvement may be ascribed to delegation of most of the computation to an RDF store in the matchmaker's back-end, whereas the previous matchmaker loaded pre-filtered subsets of data into an in-memory model, which was used to compute matches.

The metrics demonstrate that there is a room for improvement of the implemented matchmaker. As the figure 3.2 shows, an improvement may be achieved either by using a better matchmaking method or by using better data. The results suggests that especially deduplication of business entities can have a significant impact on matchmaker's performance. For example, when working with data in which every occurrence of the same business entity is given a fresh URI, it is impossible to find the correct match, as its URI occurs only once and this occurrence is withheld from the matchmaker because it is a part of the ground truth dataset.

We used the evaluation results to find better performing matchmakers and their configurations. The results suggest that the best-performing matchmaking method, both in terms of relevance and response time, is `exact-cpv` based on aggregation of exact matches of CPV codes. The performed experiments indicate that output of the matchmaking algorithm is heavily influenced by data quality in general and deduplication of business entities in particular. Further increase of matchmaking quality would thus warrant to take into consideration both the matchmaking method and quality of data that is operates on.

[15] Some studies claim that 91 % of search engine users do not consider more than the first 10 search results (http://www.seo-takeover.com/case-study-click-through-rate-google/).

[16] It was not, however, evaluated using the same rigorous setup.

4 Integration

Public Contracts Filing Application (PCFA) [3] interacts with the matchmaker's API via a proxy service that wraps the matchmaker's functionality. The application loads the matchmaker's results using asynchronous JavaScript calls that are executed on demand.

If a request to the matchmaker is done for a resource from user's private data space, the matchmaker's proxy first sends the private resource to the matchmaker using load operations described in 2.1.1 and subsequently retrieves the matchmaker's results for the resource. Otherwise, if the matchmaker is requested to retrieve matches for a resource from the public data space, it is directly provided with the resource's URI, as its representation is already available to the matchmaker, which is configured to work with the application's public data space.

Additionally, for the purposes of the PCFA, the matchmaker's proxy enhances matchmaker's results by fetching additional data, such as location of business entity. When the proxy obtains the matchmaker's results, it passes URIs of the matches into a SPARQL query template, which is then rendered and sent to the SPARQL endpoint of the application's public data space.

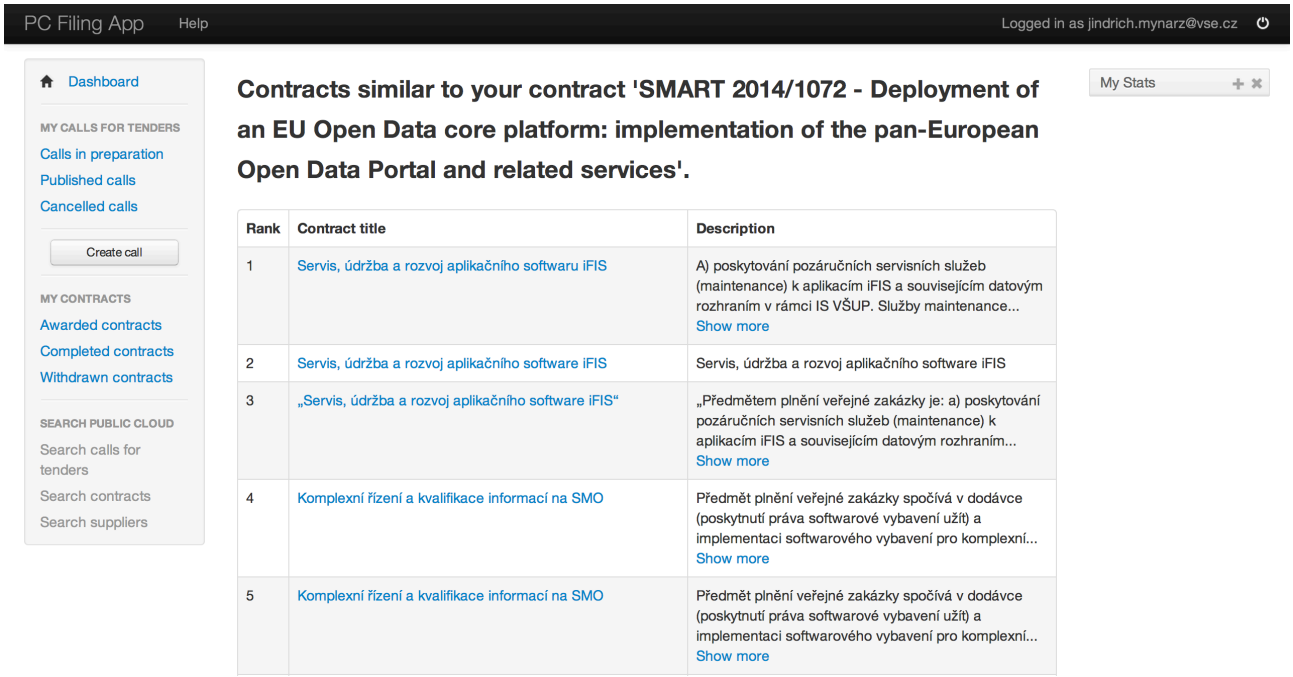

Figure 3: Similar contracts provided by the matchmaker

A representative of a contracting authority using PCFA can employ matchmaking either for unpublished private contracts or for public contracts. For both kinds of contracts a user may retrieve either similar contracts (see figure 3) or suitable suppliers (see figure 4). A user representing a bidder can employ the matchmaker to find public contracts that may be of interest to the bidder.

A demo instance of the PCFA is publicly available at http://lod2.vse.cz:8080/pc-filing-app/. Source code of the application is hosted in a public repository[17] licensed as open source under the terms of Eclipse Public License, as the matchmaker API is.

[17] https://github.com/opendatacz/pc-filing-app

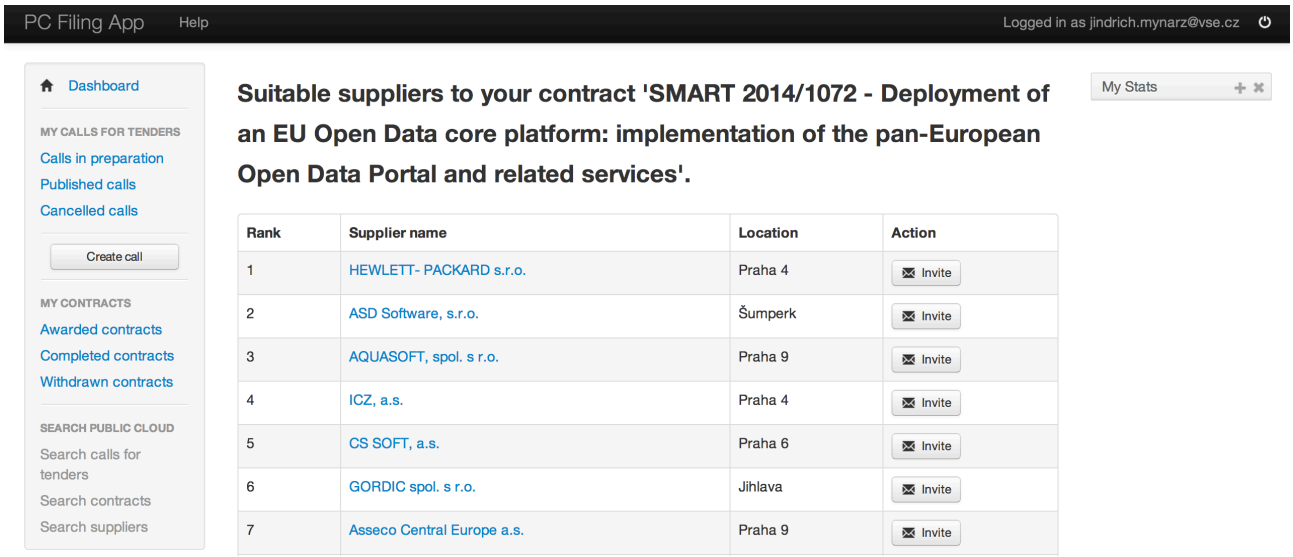

Figure 4: Suitable suppliers suggested by the matchmaker

5 Public contracts annotation with a product ontology

Matchmaking of public contracts and supplier offers can be improved by exploiting product ontologies. Both the requirements on the product or service demanded by the contractor and the specifications of the product or service offered by the supplier can be annotated provided that an appropriate product ontology for the specific industry and the detailed specification of the contract and the offers are available.

Once the products or services data is annotated there are several possible ways of exploiting it. Some of the most obvious are the automation of

- selecting offers that fulfill the contract criteria (regarding the product or service) or selecting the best offer according to the criteria
- finding similar offers and estimating an average or approximate range of prices or even other conditions (like delivery time)[18]
- selecting suppliers to be addressed according to their previous offers (i.e. finding suppliers that in the past offered the same or similar product or service demanded in the contract)

One of the biggest challenges is the annotation of the data itself. We have done some experimental work concerning annotation of Czech public contracts which we present here. A big issue is the availability of the data. The basic information about Czech contracts is publicly available in structured format on specialized web portals. However, the specification of the demanded products or services is included only in PDF attachments of the structured data. The PDF files contain unstructured plain-text documentation or sometimes even just scanned images of the documentation, making it virtually impossible to extract the product or service data automatically. Moreover, such attachments are included only sometimes and stay available only for a limited time (several months) after the contract is closed. Also, the level of detail of the product specification vary. We have chosen contracts on prescription drugs supplies for the experimental annotation as those seem to usually include quite detailed documentation.

Another issue is the availability of the product ontology for the specific industry. As the Public Contracts Ontology is used in combination with GoodRelations (GR) and GR is also supposed to serve as a basis for specific product ontologies, using such GR-based product ontology for the annotation is an obvious choice. However, such ontologies are still scarce (known to us are only several ontologies at GR website[19] and those developed in the OPDM project[20]) and none of them cover drugs. We have developed a way of extracting a Freebase schema and transforming it into GR-based product ontology and used it to obtain a prescription drug product ontology usable for the annotation.[21]

We have chosen Ontowiki as a tool for the manual annotation. An experiment with students has been done: 15 students were instructed to annotate a supplier offer included in a documentation of selected contracts. Only 47% students used GR and the drug ontology correctly and the annotation took them hours of work - partially due to some technical issues with Ontowiki - some were identified as bugs and reported.

[18] That might be used e.g., for identifying contracts were the winning tender had suspiciously high price.
[19] http://wiki.goodrelations-vocabulary.org/Cookbook
[20] http://purl.org/opdm
[21] A paper describing the transformation has been accepted to EC-Web'14 conference, the resulting ontology is available at http://pages.vse.cz/~xdudm12/ecweb2014/DON.owl.

After solving most of the Ontowiki issues, another annotation has been done by an expert. 10 public contracts with available detailed documentation including contender offers were selected and annotated. Some contracts included more than one drug and each contract had several contenders - 79 instances of `gr:Offering` were created. The experiment results suggest that both the drug ontology and Ontowiki are useful for the annotation. The annotation of each product described in the contract or offer documentation took approx. 5-10 min. of work in Ontowiki. However, finding and reading through the contract documentation have shown to be much more time consuming and we so far haven't acquired enough data to conduct matchmaking experiments with it.

6 Conclusion

The presented deliverable describes 2 software prototypes and the integration thereof. Matchmaking web services described in [4] were incorporated into the user-facing Public Contracts Filing Application (PCFA) presented in [3]. The combination of these applications was straightforward since both of them were configured to draw data from a single RDF store. The RESTful API exposed by the matchmaker was fronted with an application-specific server-side proxy for pre-processing and enriching matchmaker's results for display purposes in PCFA. The application is set up to interact with the proxy using asynchronous JavaScript calls based on user's actions.

Evaluation of matchmaker's performance was carried out using retrospective ground truth of previously awarded contracts. The evaluation results suggest that simple matchmaking based on exact matches of Common Procurement Vocabulary outperforms more complex approaches both in terms of relevance and response time. However, the matchmaking algorithm is not the only contributor to the matchmaking performance. The evaluation indicates that data quality in general, and deduplication of business entities in particular, may substantially affect matchmaking results. Richer data description, such as more assigned Common Procurement Vocabulary codes, can also contribute to improved matchmaking. This is also what detailed description of public contracts expressed via product ontologies promises. To test this hypothesis we conducted a preliminary experiment in public contracts annotation using product ontologies described in 5.

7 References

[1] Douglas, Jason; Goto, Sam; Macbeth, Steve; Johnson, Jason; Shubin, Alexander; Mika, Peter. Announcing Schema.org Actions. In *schema blog* [online]. April 16, 2014 [cit. 2014-07-03]. Available from WWW: http://blog.schema.org/2014/04/announcing-schemaorg-actions.html

[2] Lanthaler, Markus. *Hydra Core Vocabulary: a vocabulary for hypermedia-driven APIs* [online]. June 20, 2014 [cit. 2014-06-27]. Available from WWW: http://www.hydra-cg.com/spec/latest/core/

[3] Mynarz, Jindřich; Nečaský, Martin; Veselka, Jiří. *LOD2 deliverable 9a.1.2: Web application for filing public contracts* [online]. Prague, 2012 [cit. 2013-07-16]. Available from WWW: http://static.lod2.eu/Deliverables/D9a.1.2.pdf

[4] Snoha, Matěj; Klímek, Jakub; Mynarz, Jindřich. *LOD2 deliverable 9a.2.1: Prototype of matchmaking web services for linked commerce data in the domain of PSC* [online]. August 3, 2013 [cit. 2014-07-01]. Available from WWW: http://svn.aksw.org/lod2/D9a.2.1/public.pdf

[5] Sporny, Manu; Longley, Dave; Kellogg, Gregg; Lanthaler, Markus; Lindström, Niklas. *JSON-LD 1.0: a JSON-based serialization for linked data* [online]. W3C Recommendation. January 16, 2014 [cit. 2014-06-27]. Available from WWW: http://www.w3.org/TR/json-ld/

[6] Svátek, Vojtěch; Mynarz, Jindřich; Chudán, David; Klímek, Jakub; Grzybowski, Łukasz; Jarmużek, Mateusz; Węcel, Krzysztof; Bühmann, Lorenz; van der Waal, Sander. *LOD2 deliverable 9a.3.1: Application of data analytics methods of linked data in the domain of PSC* [online]. June 8, 2014 [cit. 2014-07-03]. Available from WWW: http://svn.aksw.org/lod2/D9a.3.1/public.pdf