# A Semantic Federated Search Engine For Domain-Specific Document Retrieval

Mauro Dragoni
Fondazione Bruno Kessler
Trento, Italy
dragoni@fbk.eu

Andi Rexha, Hermann Ziak, Roman Kern
Know-Center GmbH Graz, Austria
{arexha, hziak, rkern}@know-center.at

## ABSTRACT

Retrieval of domain-specific documents became attractive for the Semantic Web community due to the possibility of integrating classic Information Retrieval (IR) techniques with semantic knowledge. Unfortunately, the gap between the construction of a full semantic search engine and the possibility of exploiting a repository of ontologies covering all possible domains is far from being filled. Recent solutions focused on the aggregation of different domain-specific repositories managed by third-parties. In this paper, we present a semantic federated search engine developed in the context of the EEXCESS EU project. Through the developed platform, users are able to perform federated queries over repositories in a transparent way, i.e. without knowing how their original queries are transformed before being actually submitted. The platform implements a facility for plugging new repositories and for creating, with the support of general purpose knowledge bases, knowledge graphs describing the content of each connected repository. Such knowledge graphs are then exploited for enriching queries performed by users.

## CCS Concepts

**•Information systems → Web services; Information retrieval query processing; Combination, fusion and federated search;**

## Keywords

Information Retrieval, Semantic Web, Federated Search, Knowledge-based Query Expansion

## 1. INTRODUCTION

The adoption of semantic approaches in the field of information retrieval becoming quite common in the last decade. This research direction has been supported by the development of tools and resources fostering the creation of information layers containing semantic structured data coming from external resources. In particular, the creation of ontologies, knowledge bases, and algorithms fostering the annotation of natural language documents by exploiting conceptual knowledge, pushed the development of semantic information retrieval systems [1]. From a research perspective, several works demonstrated the suitability of these approaches [13]. Unfortunately, some limitations occur when these solutions have to be integrated within real-world information retrieval systems. In case the requirement is the implementation of a single-domain retrieval system, the implementation can be done by integrating domain-specific ontologies and ad-hoc annotation tools. On the contrary, the implementation of a general purpose solution suffers from the lack of knowledge for describing, with a high level of detail, all possible domains that can be contained in the retrieval system.

In this paper, we present a solution placed in the middle of these scenarios. Our goal is to develop an architecture able to connect, in a federated way, several domain-specific retrieval services managed by third-parties. This platform has been created within the EEXCESS EU-project [1], where it was required the implementation of a semantic-bases search engine supporting the possibility of expanding the retrieval capability of the system to several domains without affecting the efficiency and the effectiveness of the entire system. The core module is the federated search component which roles are: (i) given a query, it is in charge of detecting the domain of the query and, consequently, to enrich it with semantic information coming from knowledge graphs describing that domain; and (ii) to operate without affecting the general performance of the system in order to hide the distributed retrieval activity to users.

The paper is structured as follows. Section 2 contains a brief overview of federated search platform deployed in real-world environments. Section 3 describes the general architecture on the system and presents the main components involved in the management of the federated semantic search. In Section 4, we evaluate the impact of the federated search component within the platform and, finally, Section 5 concludes the paper.

## 2. RELATED WORK

Federated search is the task of distributing a query to multiple search engines and combining results into a single, consolidated search result. Thus, for users it appears as their were interacting with a single search instance. Such techniques originated in the field of Web search, often aiming of improving the quality and to reduce unwanted content [3]. Here multiple Web search engines are combined for a single result list, where the results typically represent

*SAC 2017,April 03-07, 2017, Marrakech, Morocco*
Copyright 2017 ACM 978-1-4503-4486-9/17/04. . . $15.00
http://dx.doi.org/10.1145/3019612.3019833

[1]All information about the project can be found at http://eexcess.eu/

Web pages. In such a scenario the results are homogeneous, sharing the same characteristics and available meta-data.

The term federated search has become increasingly popular to describe systems, that search over multiple document collection simultaneously [12]. Here the fact, that these document collections are stored separately is stressed. In cases, where the type of results from different search engines differ, the term vertical search is commonly used. Due to the different nature of the results, the individual result lists are no longer aggregated, but the results from the different underlying search engines are display separately. For example, there might be relevant images, as well as textual document for a given query, which are appear in different parts of the user interface.

The term aggregated search is used to indicate that multiple search results are combined into a single one [6], even if the results might be generated from sources of varying types. Such systems are also called multi domain meta search engines [9]. In such a context one can distinguish between a so called cooperative setting and an uncooperative setting. In the latter case the underlying search engines cannot be modified and their corpora are not accessible from the outside. The EEXCESS federated search component falls into this categories, as the behaviour of the connected services can only be steered via manipulating the query.

According to the literature [7], the three main challenges that aggregated search faces are: i) the selection of the appropriate sources that best represent the user's information need, ii) the so called collection representation problem, i.e. inferring the key characteristics of a knowledge base while keeping the effort minimal iii) the aggregation of the results returned by the different sources.

An important aspect in an uncooperative setting is the query processing aspect. Here problems may arise due to short queries, where the user information need is represented only by few words. For such short queries one approach is to expand the query via related terms [10]. At the opposite end of the spectrum are queries consisting of many terms. In such case queries are long and may consists of multiple distinctive concepts. Hence the idea is to split a single query into segment, which are themselves coherent [4].

## 3. PLATFORM ARCHITECTURE

In this section, we present the architecture implemented in the EEXCESS project specifically addressing the federated search aspect. A diagram of such an architecture is shown in Figure 1.

In this Section, we describe how we have supported the two main processes shown in Figure 1: (i) the connection of a new service to the platform with the creation of the knowledge graph describing its content, and (ii) the management of the queries performed by users, from the enrichment of the provided natural language text, to the aggregation of the results provided by the services actually queried.

Below, we provide the description of these two processes by highlighting the role of Semantic Web technologies in achieving the goal of creating an effective, and also efficient, semantic federated search engine.

### 3.1 Service Connection and Knowledge Graphs Building

The connection of a new retrieval service consists in analyzing the documents contained in its repository and to create a knowledge graph describing its content. The created knowledge graphs are used for supporting the detection of the domain targeted by the queries provided by the users and for enriching performed queries with the aim of improving the effectiveness of the retrieval task. Each knowledge graph contains a set of interconnected knowledge objects that, in turn, are composed by two layers: (i) a linguistic layer built from the key terms extracted from a document (lemma, part-of-speech, multilinguality, etc.), and (ii) a semantic layer containing structured data, represented with triples, related to named entities associated with the linguistic terms, relationships of interest for that particular key term, etc. These facilities are placed on the right side of Figure 1 and grouped under the "Finger Printer" package.

#### *3.1.1 Libraries*

The "Finger Printer" includes two libraries that are used also in other components of the platform.

The first library is the **C4** [2] (SCientific and Cultural Content in Context) one. C4 has been developed with goals broader than the sole extraction of key terms from sentences or paragraphs of text. Indeed, the full capabilities of C4 include also the disambiguation task and the log of users while they perform queries by supporting the construction of their interests profiles. Concerning the platform presented in this paper, the C4 library has been used for supporting the construction of the language model describing the content of each service repository, and for associating users' queries with the right domain.

The second library is **DoSeR** [3] (Disambiguation of Semantic Resources), used for named entity disambiguation and category assignment. Given a natural language text and a knowledge base, DoSeR creates annotations by creating links between natural language expressions and entities defined in the knowledge base.

Beside the two libraries described above, others, that are out of the scope of this paper, have been developed and can be found on the Github repository of the project [4].

#### *3.1.2 Metadata Extraction and Named Entity Annotation*

Once a service is connected to the platform, the first performed task is the construction of the knowledge graph in order to profile the content of the documents contained in the service repository.

In analyzing the connected repository, we might encounter two possible scenarios: (i) the service provides documents that have been annotated early, therefore, from such annotations we are already able to build the knowledge graph, or (ii) the service provides documents in raw format only and they have to be annotated before to build the knowledge graph.

The first scenario is not very common due to the effort required by service curators for managing all annotations. However, in some cases documents can be automatically (or exceptionally manually) annotated by exploiting domain-specific ontologies or metadata schemata. An example is the medical domain, where papers published on the PubMed [5] portal are always annotated by the authors with concepts

[2]https://github.com/EEXCESS/c4
[3]http://purl.org/eexcess/components/research/doser
[4]https://github.com/EEXCESS/
[5]http://www.ncbi.nlm.nih.gov/pubmed

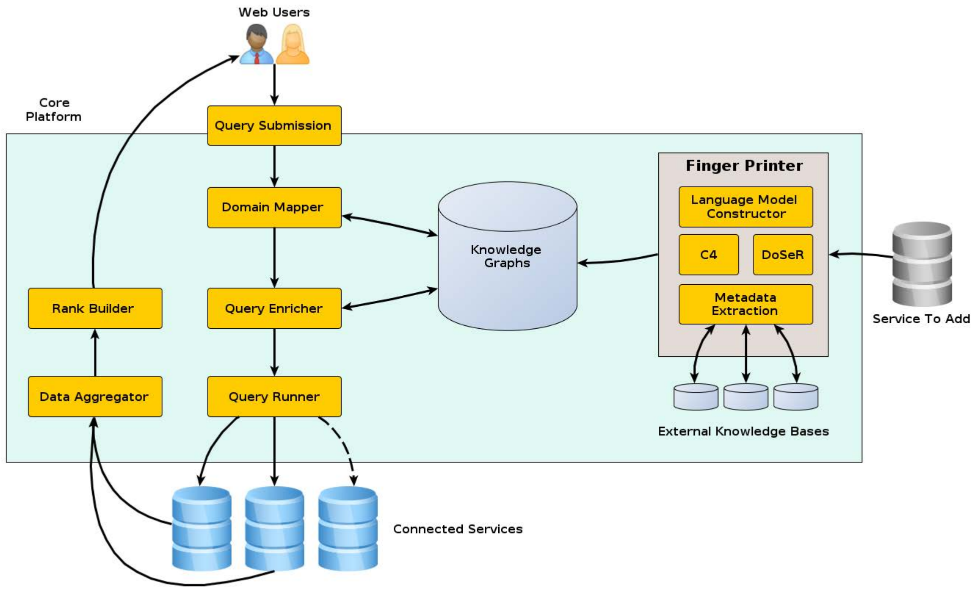

**Figure 1: The diagram of the architecture implemented for the EEXCESS project.**

contained in the UMLS Knowledge Base [6]. In this case, each annotation is automatically translated in a knowledge object and inserted into the knowledge graph.

In the second scenario documents are provided in a natural language format without annotations. In this case, knowledge bases (general purpose or domain specific) are exploited for creating annotations. Given the content of a document, the "Metadata Extractor" module analyzes the text, extracts concepts and entities by using the DoSeR library, and defines the links between such entities and the ones contained in used knowledge bases. In the EEXCESS project implementation, the "Metadata Extractor" component exploits general purpose knowledge bases like DBpedia [7], Freebase [8], and YAGO [9]; while the use of domain-specific knowledge bases has been planned as future work for extending platform functionalities.

As mentioned early, we adopted the disambiguation algorithm contained in the DoSeR library. The evaluation of the DoSeR library provided in [14] reveals that the strategy integrated into DoSeR attained the best disambiguation results.

The result of this task is a set of knowledge objects, connected between them, injected into the knowledge graph. Each knowledge object includes: (i) the labels of the entity identified in the text; (ii) a weight computed by considering the frequency of the entity in the entire service repository; and (iii) a set of relationships with further entities or data.

#### *3.1.3 Language Model Construction*

This task is in charge of building the linguistic layer of knowledge objects. The linguistic layer is exploited during query execution for computing the "belonging degree" of a query to a particular domain.

The building process is done in two phases. In the first phase, we apply a keyword extraction algorithm for detecting the most significant terms contained in each document.

The extraction of the keywords from a given text is demanded to an algorithm integrating a Doc2vec-based approach. Generally based on Word2Vec [11], Doc2Vec produces a vector given a sentence or a document. For each paragraph contained in a document, we infer a representative vector given a Doc2Vec model created on a corpus containing all Wikipedia pages. We compare this vector with the ones computed on each Wikipedia page by adopting the cosine similarity measure. The Wikipedia page with the highest similarity with the input paragraph represents the main topic of the paragraph. From the two vectors, we extract the matching keywords and we create, for each keyword, a new node in the knowledge graph. The creation of the node is done only if such a node does not exist yet. The node will contain the keyword itself and the set of its translations, where available.

After this step, we perform a further keyword extraction task with the aim of extracting general categories describing document content. In the previous subsection, we stated that we annotate the entities found in each documents. From such annotated entities, we are able to create statistics of categories associated with them. After the extraction of all categories, we create a distribution given all categories of the identified entities. The top five categories are inserted in the knowledge graph by creating a dedicated knowledge object like happens for keywords. The rationale behind the choice of the top five categories is to have a trade-off between inserting categories that are distinctive with respect to the document content and avoiding too much noise given categories that are less significant.

## 3.2 Query Manager

The second supported process is the execution of queries over the

[6]https://www.nlm.nih.gov/research/umls/
[7]http://wiki.dbpedia.org/
[8]https://www.freebase.com/
[9]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

services that are connected to the platform. Here, users provide natural language queries that are executed in a distributed way by keeping the tasks of domain detection, enrichment, federated execution, and results aggregation completely transparent from the user perspective. The pipeline of the query manager process is reported on the left side of Figure 1.

The aim of this process is to infer the context of each query and to enrich them before giving them as input to the federated search component. In accordance with the user-based information seeking model presented in [8], the process follows these steps: (i) to identify the relevant context; (ii) to recognize the information need; and (iii) to express this information need in terms of an enriched query given as input to the federated search component.

### 3.2.1 *Domain Mapper*

When a new query is performed, the first step consists in detecting the query domain in order to redirect the system to the right repository for executing it.

Achieving high classification accuracy on the domain detection task often relies on expressive representations for words. Word2Vec word representations, estimated from large text corpora, improve the accuracy on many natural language processing tasks through their high-quality features. As described early in this Section, this kind of approach has been used for creating the language model describing each domain.

Here, we implemented a domain-matching approach based on the similarity between the terms contained in the query and the linguistic layer of the knowledge objects contained in knowledge graphs. For each query term, we check if a knowledge object containing such a term as label exists. If yes, the knowledge object is "turned on"; otherwise, the algorithm looks for nodes containing synonyms, or related terms, of the query term. We perform this operation for all terms contained in the query on all knowledge graphs (one for each repository connected to the platform) stored in the system. At the end of this matching operation, relationships between the query and one or more domains are created by checking how many knowledge objects have been turned on by the query terms.

It is important to highlight two aspects: (i) the system may be connected to different retrieval services belonging to the same domain, and that (ii) more than one domains would be selected. In both cases, the implemented strategy consists in exploiting the similarity values as weights used for multiplying scores computed on the retrieved documents. A more in depth analysis of the implemented approach and the design of possible alternatives are part of future work.

### 3.2.2 *Query Enricher*

The second step consists in the enrichment of the performed queries. The enrichment task exploits the output of the domain mapping by using only the knowledge graphs belonging to the selected domain. Above, we mentioned that knowledge objects contained in the knowledge graph of the mapped domain are "turned on" when they match with the content (or meaning) of query terms. By starting from such knowledge objects, we build different information layers that are used for representing the final query sent to the select services. In particular, the query will be composed by the following layers:

- Keyword layer: contains the original terms of the query provided by users.
- Conceptual layer: contains labels of the concepts that are directly connected with the knowledge object of the original terms.
- Entity layer: contains the named entities detected in the query (by using the DoSeR library) or that are directly connected with the knowledge object of the original terms.
- Related information layer: other information that are directly connected with the concepts used for populating the Conceptual layer.

These information are stored in a JSON object for keeping layers separated. This way, it will be possible to easily manage the query submissions; for instance, a retrieval service may support a multi-layer query parsing or the possibility of managing and scoring the layers in different ways.

### 3.2.3 *Query Runner and Results Presentation*

This last component is in charge of performing enriched queries for retrieving documents from the services belonging to the domain detected by the "Domain Mapper" component. At this stage is important to highlight that each retrieval service is managed independently by its service provider and it may implement different scoring techniques.

Indeed, here, the main drawback of the designed architecture emerged. Unfortunately, it is the impossibility to have a direct control on the techniques used for computing the relevance of documents with respect to user queries.

The scoring function used by each service for computing document relevance are seen as black boxes by the federated search component. For this reason, it has been implemented a mechanism, based on the one presented in [2], for aggregating and, eventually, reranking, the results provided by each service before to provide a unique result to users. Briefly, such a mechanism normalizes the scores computed on each service and multiplies them by a weight proportional to the belonging degree of the original query to the domain used services belong to.

## 4. SYSTEM EVALUATION

The evaluation of the system has been done in two steps: (i) the first one provides an evaluation of the scalability and the robustness of the federated search service; while (ii) the second one presents measures, in terms of precision and recall, of the effectiveness of the enriched queries. The latter aims to validate the contribution of information extracted from knowledge graphs in the retrieval activity.

Since scalability and perceived performance of the system are important factors for the acceptance by users, we conducted tests to evaluate the overall system performance and find potential bottlenecks in the system architecture. We designed four settings for testing system performance and for measuring the impact of the federated search component.

- Setting 1: on the same server, we installed the system and a service containing 150,000 documents coming from Wikipedia; here, the federated search component is disabled.

- Setting 2: on the same server, we installed the system and three services containing 50,000 documents each, coming from the collections of three partners: Mendeley, Europeana, and KimPortal; here, the federated search component is enabled.
- Setting 3: on two servers (placed in two different locations), we installed the system on server one and a service containing 150,000 documents coming from Wikipedia on server two; here, the federated search component is disabled.
- Setting 4: on four servers (placed in four different locations), we installed the system on server one and three services (on server two, three, and four respectively) containing 50,000 documents each coming from the collections of three partners: Mendeley, Europeana, and KimPortal; here, the federated search component is enabled.

Each run consisted of 1649 unique queries that were sent to the system in blocks from 10 to 500 queries at a time with 10 second breaks between the blocks to let the system recover from the load. These settings simulated the behavior of several users, who query the retrieval service simultaneously within a few milliseconds. Results of run executed under Settings 1 and 2 are shown in Figure 2 and 3 respectively.

Both figures show that timeouts start to appear at around 50 and 100 parallel calls, depending on the setup. For Setting 2, the system errors caused by the federated search component are around 0.35% at 500 parallel calls. From the graphs, we may infer that the main timing component is provided by the retrieval services that, with increasing load, yield higher latencies, which, at a certain stage, provoke timeouts. By comparing the "Time" data for Settings 1 and 2, the federated search component does not appear to play the role of bottleneck and it copes well with load increasing.

A second set of runs was conducted by simulating the usage of multiple distributes retrieval services deployed in different geographical locations. As mentioned early, given four servers, the federated search component was running on the first server, while the retrieval services on the other three. This setup allowed to observe how the federated search component coped with the geographical distribution across several machines and which issues might be encountered. Figure 4 shows the results obtained by running Setting 3. Here, it is possible to observe a slightly performance improvement related to the service timeouts.

This aspect is confirmed also from the running of Setting 4 (Figure 5). Here, the failure rate of the federated search component at 500 parallel queries drops to 0.3%. This positive improvement is given due to the distribution of multiple search systems on different servers with respect to Setting 2 where only one server was used.

The peak load of the system, when the federated search component was running, never exceeds 30% per CPU, where the average load was around 10%. This again corroborates that the scalability of federated search component is good independently from how many users can be simultaneously served.

The second quantitative evaluation aims to validate the effectiveness of the enriched queries on the ranks presented to users. The evaluation procedure has been performed in a closed environment [5] with the following setup:

- the set of documents contained 400,000 articles coming from all repositories;

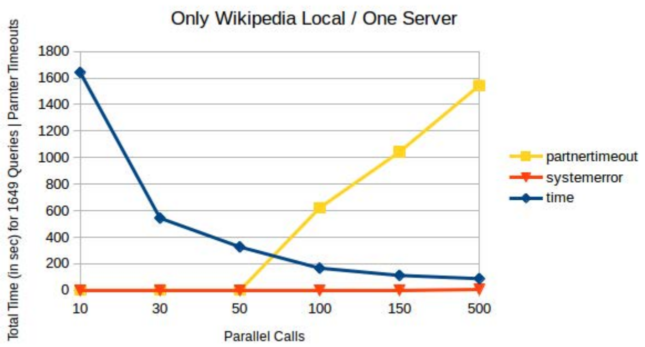

Figure 2: Runtime behavior obtained by applying Setting 1.

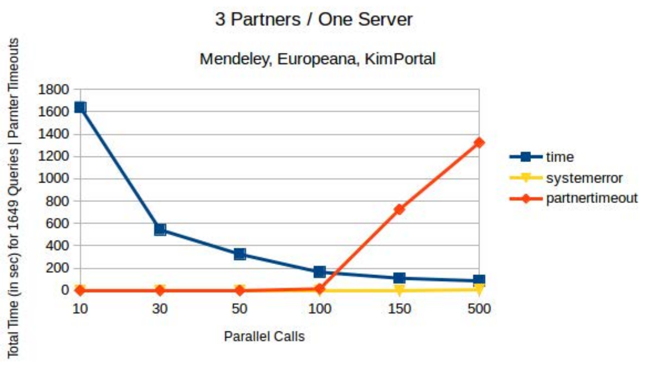

Figure 3: Runtime behavior obtained by applying Setting 2.

- the set of queries contained 60 queries assigned to 30 different users coming from outside the project context;
- as baselines, (i) we manually enriched the queries by assuming that the resulting query is the "optimal" one that can be performed by starting from the ones assigned to users, and (ii) we measured the effectiveness of the system without enriching queries with the contribution of knowledge graphs;
- the rank composed by the first 10 retrieved documents have been evaluated;
- all ranks have been validated by experts;
- precision, recall, and f-measure have been measured at the 10th position.

| | Manual Enrichment (Baseline) | Automatic Enrichment | No Enrichment |
|---|---|---|---|
| Precision | 0.81 | 0.78 | 0.65 |
| Recall | 0.72 | 0.63 | 0.54 |
| F-Measure | 0.76 | 0.69 | 0.59 |

Table 1: Effectiveness validation of the enriched queries.

Table 1 shows the results computed by comparing the automatic enriched queries with the two baselines. By assuming that the "optimal" enrichment achieves the best possible result, by applying the automatic enriched queries, we obtained a performance equal

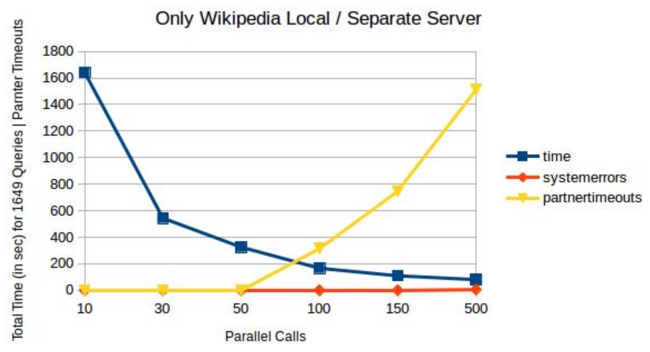

Figure 4: Runtime behavior obtained by applying Setting 3.

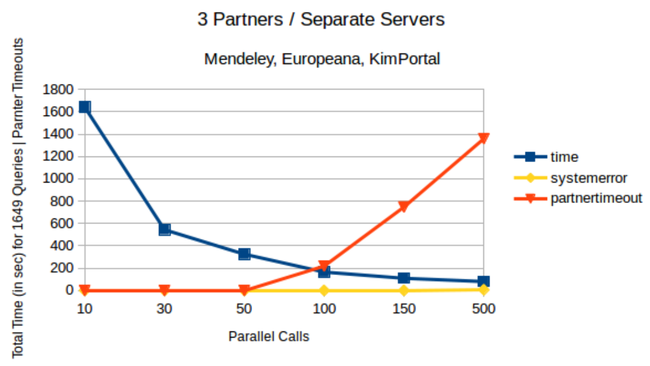

Figure 5: Runtime behavior obtained by applying Setting 4.

to 91% (0.69/0.76) of the "optimal" one. While, by comparing the results with the no-enriched queries, the performance gain rose to 117%. These results are encouraging for future improvement of the algorithm used for automatically enriching queries by exploiting knowledge graphs.

## 5. CONCLUSIONS

In this paper, we presented the platform implemented in the EEXCESS EU-project for supporting the task of semantic federated document retrieval. The platform allows to connect domain-specific retrieval services in order to exploit semantic resources for enriching user queries. The core component of the platform, the federated search component, (i) maps users' queries to one of the domain supported by the retrieval services, (ii) exploits the content of the related knowledge graph for enriching such queries, and (iii) sends the enriched queries to the proper services. The system has been evaluated from an efficiency perspective demonstrating the almost absent impact of the federated search component on the general performance of the platform independently of the tested settings. The evaluation performed on the produced ranks demonstrated the quality of the semantic enrichment of the queries.

## 6. REFERENCES

[1] F. Corcoglioniti, M. Dragoni, M. Rospocher, and A. P. Aprosio. Knowledge extraction for information retrieval. In *The Semantic Web. Latest Advances and New Domains - 13th European Semantic Web Conference, ESWC 2016, Creete, Grecia, May 29 - June 2, 2016. Proceedings. To appear.*, 2016.

[2] C. da Costa Pereira, M. Dragoni, and G. Pasi. Multidimensional relevance: Prioritized aggregation in a personalized information retrieval setting. *Information processing & management*, 48(2):340–357, 2012.

[3] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001.

[4] M. Hagen, M. Potthast, A. Beyer, and B. Stein. Towards optimum query segmentation: In doubt without. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1015–1024, New York, NY, USA, 2012. ACM.

[5] P. Ingwersen and K. Järvelin. *The Turn - Integration of Information Seeking and Retrieval in Context*, volume 18 of *The Information Retrieval Series*. Springer, 2005.

[6] A. Kopliku, K. Pinel-Sauvagnat, and M. Boughanem. Aggregated search: A new information retrieval paradigm. *ACM Computing Surveys (CSUR)*, 46(3):41, 2014.

[7] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Advances in Information Retrieval*, pages 52–66. Springer, 2005.

[8] G. Marchionini and R. White. Find what you need, understand what you find. *Int. J. Hum. Comput. Interaction*, 23(3):205–237, 2007.

[9] D. Minnie and S. Srinivasan. Meta search engines for information retrieval on multiple domains. In *Proceedings of the International Joint Journal Conference on Engineering and Technology (IJJCET 2011)*, pages 115–118. Citeseer, 2011.

[10] J. Montgomery, L. Si, J. Callan, and D. A. Evans. Effect of varying number of documents in blind feedback: analysis of the 2003 nrrc ria workshop bf_numdocs experiment suite. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 476–477. ACM, 2004.

[11] X. Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

[12] M. Shokouhi and L. Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.

[13] N. Stojanovic. An approach for the efficient retrieval in ontology-enhanced information portals. In D. Karagiannis and U. Reimer, editors, *Practical Aspects of Knowledge Management, 5th International Conference, PAKM 2004, Vienna, Austria, December 2-3, 2004, Proceedings*, volume 3336 of *Lecture Notes in Computer Science*, pages 414–424. Springer, 2004.

[14] S. Zwicklbauer, C. Seifert, and M. Granitzer. From general to specialized domain: Analyzing three crucial problems of biomedical entity disambiguation. In Q. Chen, A. Hameurlain, F. Toumani, R. Wagner, and H. Decker, editors, *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I*, volume 9261 of *Lecture Notes in Computer Science*, pages 76–93. Springer, 2015.