D3.6 Semantic Digital Interlinking and Clustering Prototype v2

DURAARK

FP7 – ICT – Digital Preservation
Grant agreement No.: 600908

Date: 2015-07-31
Version 1.0
Document id. : duraark/2015/D.3.6/v1.0

Grant agreement number	:	600908
Project acronym	:	DURAARK
Project full title	:	Durable Architectural Knowledge
Project's website	:	www.duraark.eu
Partners	:	LUH – Gottfried Wilhelm Leibniz Universitaet Hannover (Coordinator) [DE] UBO – Rheinische Friedrich-Wilhelms-Universitaet Bonn [DE] FhA – Fraunhofer Austria Research GmbH [AT] TUE – Technische Universiteit Eindhoven [NL] CITA – Kunstakademiets Arkitektskole [DK] LTU – Lulea Tekniska Universitet [SE] Catenda – Catenda AS [NO]
Project instrument	:	EU FP7 Collaborative Project
Project thematic priority	:	Information and Communication Technologies (ICT) Digital Preservation
Project start date	:	2013-02-01
Project duration	:	36 months
Document number	:	duraark/2015/D.3.6
Title of document	:	Semantic Digital Interlinking and Clustering Prototype - II
Deliverable type	:	Software prototype
Contractual date of delivery	:	2015-07-31
Actual date of delivery	:	2015-07-31
Lead beneficiary	:	LUH
Author(s)	:	Jakob Beetz `<j.beetz@tue.nl>` (TUE) Ujwal Gadiraju `<gadiraju@l3s.de>` (L3S) Besnik Fetahu `<fetahu@l3s.de>` (L3S) Stefan Dietze `<dietze@l3s.de>` (L3S) Martin Hecher `<dietze@l3s.de>` (L3S)
Responsible editor(s)	:	Besnik Fetahu, Ujwal Gadiraju, Stefan Dietze (LUH)
Quality assessor(s)	:	Sebastian Ochmann `<ochmann@cs.uni-bonn.de>` Michelle Lindlar `<Michelle.Lindlar@tib.uni-hannover.de>`
Approval of this deliverable	:	Jakob Beetz `<j.beetz@tue.nl>` (TUE) Stefan Dietze `<dietze@l3s.de>` (L3S)
Distribution	:	Public
Keywords list	:	Semantic Enrichment, Interlinking, Crawling, Clustering, Linked Data

Executive Summary

This deliverable describes the second version of the interlinking and clustering prototype, where significant improvements have been designed and implemented. The previously developed SDO (D3.2) provides the metadata (profiles) corresponding to available datasets to be crawled. It also facilitates the detection of targeted entry points into the LOD graph for specific seed lists. While this functionality is deemed essential for scenarios where a wide range of datasets are meant to be archived/crawled, the narrow DURAARK domain (architecture) limits the potentially relevant datasets to a reasonable amount. To this end, SDO data is provides a sound foundation for pre-configuring the focused crawler and cannot be used dynamically for looking up datasets for each crawl. In this deliverable, we introduce a focused crawler for linked data (LD), which extends the previously developed crawling environment with a more targeted and hence scalable approach. Crawls are either based on (a) manually defined seed lists, for instance, to retrieve relevant LD subgraphs about the geographic, historical or infrastructural context of buildings and their model or (b) automatically extracted seeds, directly derived from existing buildM instances. Based on experimentally defined crawl configurations, we introduce an efficient means to crawl linked data of relevance to the specific instances in the SDA (Semantic Digital Archive). The focused crawler is exposed as DURAARK micro service (D2.5) and already integrated into the DURAARK WorkbenchUI and workflows. In addition, we introduce an entity retrieval approach which extends existing state of the art methods and provides improved retrieval performance. Given the limited amount of data in the SDA, current experiments have been carried out on top of large-scale datasets, namely the billion triple challenge dataset – BTC [1]. As part of future work and evaluation activities in DURAARK, this work will be applied as part of the SDA search and retrieval workflows.

[1]http://km.aifb.kit.edu/projects/btc-2014/

Table of Contents

1 Introduction

This document describes the second version of the interlinking and clustering prototype, where significant improvements to the previous version (D3.4) have been implemented. In particular, we introduce a focused crawler for linked data, which replaces the previously developed crawling environment with a more targeted and hence scalable approach. Based on experimentally defined crawl configurations, we introduce an efficient means to crawl linked data of relevance to the specific instances in the SDA. In addition, we introduce an entity retrieval approach which extends existing state of the art methods and provides improved retrieval performance. While the crawler is already an integral part of the DURAARK WorkbenchUI and integrated into the archival and enrichment workflows, the entity retrieval approach will be applied as part of the SDA search and retrieval workflows in future work.

1.1 Moving Further From D3.4

The main contributions in this work are the following:

- We introduce and implement a new approach to focused crawling for semantic enrichment that is more targeted and scalable.
- We integrate the focused crawler into the DURAARK WorkbenchUI and workflows. In addition, we expose this as a micro-service.
- We implement a novel approach for clustering and retrieval of entities, in the context of searching data in the SDA
- Essential evaluations have been carried out on large-scale datasets to assess and validate performance for the focused crawling and the clustering and retrieval approaches.

1.2 Addressed Use Cases

The two software components introduced in this deliverable are integrated into the overall DURAARK system (D2.5) and to the use cases and workflows described in D2.5. The original use cases defined for DURAARK (D2.1 - Requirements Document) are summarised below.

The use cases related to *core long-term preservation* tasks are:

- UC1: Deposit 3D architectural objects
- UC2: Search and retrieve archived objects
- UC3: Maintain Semantic Digital Archive

The use cases related to *production and consumption-oriented* tasks are:

- UC4: Detect differences between planning state and as-built state
- UC5: Monitor the evolution of a structure over time
- UC6: Identify similar objects within a point-cloud scan
- UC7: Plan, document and verify retrofitting/energy renovations
- UC8: Exploit contextual information for urban planning
- UC9: Enrich BIM/IFC model with metadata from a repository

As part of recent integration and orchestration work, use cases and microservices have been structured into three key *workflows*, which directly involve these aforementioned use cases:

- **Archival/Input** workflow
- **Retrieval/Output** workflow
- **SDA Maintenance** workflow

The two components and microservices introduced in this deliverable, *focused crawling (FC)* and *entity retrieval (ER)*, are directly involved in all three workflows and a variety of use cases. Table 1 provides an overview of workflows, use cases and the involvement of the two components discussed here.
The components shown in Table 1, namely the FC and ER components are implicitly or explicitly contributing to the majority of use cases, where only UC4 is not benefiting from the functionalities described here. We have indicated an *explicit* contribution, if the corresponding micro-service is directly invoked during the corresponding use case while

an *implicit* contribution is indicated in cases, where earlier invocations have facilitated or added to the use case.
Let us consider for instance *UC2 - search and retrieve archived objects.* Here, the ER component directly contributes to the SDA search facilities (explicit contribution), while the FC component implicitly contributes, since search is facilitated by semantic enrichments provided through the focused crawls.
Additional examples on how enrichments/crawls facilitate the user queries identified in WP7 are provided in Section 3.3.

1.3 Overview of this Deliverable

In D3.4, delivered in M21, we introduced the general concepts and initial prototypes of interlinking and clustering, alongside the approaches taken in the prototypical software tools to be implemented as part of the DURAARK system. We also discussed results of preliminary experiments.
In this deliverable, we elucidate our progress with respect to (i) focused crawling as a means for semantic enrichment of archival data (see Section 2), and (ii) clustering methods and entity retrieval in Section 3.2. This deliverable extends the D3.4 additional practical applications of these prototypical tools, in particular for the focused crawling component. Based on larger scale experiments, the prototypes will be tested and evaluated on additional datasets by exploiting controlled empirical experiments in the lab and crowdsourcing within the context of evaluation initiatives of WP7.

Worfklow	Use Case	FC	ER	Notes
Archival	UC1	explicit	none	FC directly facilitates enrichment during ingest and archival.
	UC9	explicit	none	FC directly facilitates enrichment during ingest and archival. Semantic enrichment considers relevant Linked Data from variety of datasets/repositories.
Retrieval	UC2	implicit	explicit	ER directly contributes to the retrieval of model metadata in the SDA. FC implicitly impacts the retrieval by providing additional semantic metadata and context information.
	UC4	none	none	Difference detection is affected by geometric enrichment (WP4/5) components only.
	UC5	implicit	explicit	Metadata and contextual knowledge crawled from available LD sources can enable the tracking of the evolution of a structure or its context, for instance, its geographic, infrastructural or environmental context.
	UC6	implicit	explicit	Similarity of objects is defined both through their geometric features (floor count, size etc) but also features of their context (for instance, location, population of city/country, age, regional political legislations relating to built environment). To this end, semantic enrichment as covered by FC and ER components contributes to UC6.
	UC7	explicit	explicit	During retrofitting scenarios, contextual knowledge about the surrounding infrastructure, environment or the legal setting are crucial information, captured by the FC component and retrieved through ER.
	UC8	explicit	explicit	Contextual knowledge is provided by the FC and retrieved through the ER component.
SDA Maint.	UC3	implicit	implicit	Both FC and ER are inherent elements of the SDA infrastructure, facilitating the population and search of the SDA and as such, are inherent to SDA maintenance.

Table 1: Workflows and use cases addressed through focused crawler (FC) and entity retrieval (ER) component.

1.4 Accessing the Software Prototypes

The two software prototypes - *Focused Crawler*, *Entity Retrieval* - described in this deliverable can be accessed by different means. Both components are available as source code via the URLs proposed below. In addition, while the crawler is available both as DURAARK micro service as well as already integrated into the DURAARK WorkbenchUI, we describe its public REST APIs and the user interface as part of the work bench.
Authorization. In order to access the focused crawling REST API, we have set up an HTTP authentication mechanism. The credentials for the REST API are the following: `username:` *duraark* and `password:` *duraark_services*.

1.4.1 Source Code

Focused Crawling: The source code is accessible at the following url[2]. It is an integration of candidates crawling, ranking and REST API[3].
Entity Retrieval/Clustering: The source code is publicly available at the following url[4]

1.4.2 REST-ful APIs

The focused crawler described in this deliverable provides a REST API through which basic functionalities are exposed as HTTP GET methods. The API allows to trigger crawls or retrieve/monitor crawls and is integrated into the DURAARK WorkbenchUI. While the API has been specifically developed to enable integration into the DURAARK scenarios and use cases, it is a public Web service and enables usage of the crawler by third parties, seeking to retrieve targeted subgraphs of public Linked Data. For the detailed implementation of the focused crawling functionalities on the publicly accessible REST API, we refer to Section 4.

1.4.3 WorkbenchUI

The crawler is integrated into the DURAARK prototype in two ways:

1. Semantic enrichment of SDAS data ("buildM instances") as backend process when populating the SDA.

[2] https://github.com/bfetahu/focused_crawler
[3] http://data.duraark.eu/services/CrawlAPI
[4] https://github.com/bfetahu/entity_clustering

2. User-triggered crawls through the DURAARK WorkbenchUI.

While the former is implemented as backend functionality (exploiting the basic crawling functionalities) it facilitates the gradual enrichment of metadata in the SDA - about physical assets and their digital models - with related contextual information. Here, seeds are extracted from new data (for instance, the geolocations of physical assets) which serve as starting point for retrieving contextual information.
Further, user-triggered crawls can be initialised through the DURAARK WorkbenchUI, for instance, in order to retrieve targeted contextual data about specific entities, e, geolocations of specific importance to the data in the SDAS. User-triggered crawls are supported through so-called crawl templates, which already realise a specific crawl intent, for instance, to retrieve geographical, historical, or otherwise contextually related data.
The integration of the focused crawling into the DURAARK WorkbenchUI is described in Section 4.

2 Focused Crawling

In this chapter, we describe an adaptive and targeted approach to focused crawling. We first present a thorough description of our approach and experiments. Then we discuss how our generic approach can be applied to the specific context of DURAARK, and finally provide excerpts of the data thus generated. Further details of the approach described below are available in [24].

Given the evolution of the Linked Data, crawling and preservation have become increasingly important challenges. Due to the scale of available data, efficient focused crawling approaches which are able to capture the relevant semantic neighborhood of seed entities are required. Here, determining relevant entities for a given set of seed entities is a crucial challenge. While the weights of seeds within a seed list vary significantly with respect to the crawl intent, we argue that an adaptive crawler is required, which considers such characteristics when configuring the crawling and relevance detection approach. To address this problem, we introduce a crawling configuration, which considers seed list features and evaluate it through extensive experiments in comparison to a number of baseline methods and crawling parameters. We demonstrate that configurations which consider seed list features outperform the baselines and present further insights gained from our experiments.

2.1 Crawling to Populate a Domain-Specific Knowledge Base

This section is structured as follows. We first introduce and motivate the importance of our current work. Next we discuss related literature, followed by a problem definition and overview of the approach. Section 2.3 describes the seed list analysis, which provides a foundation for the adaptive crawling described later in the same Section. The experimental setup and results are described in Section 2.5 and 2.6 respectively. We discuss and conclude our work in the context of adaptive focused crawling in Section 2.8.

2.1.1 Introduction

Crawling and preservation of entities is an important challenge. Usually entities as part of Linked Datasets are interlinked with other entities into a semantic neighborhood, with links having specific semantics. However, entities and their semantic neighborhood evolve, which leads to a number of practical implications.

For instance, we consider the task of document annotation, namely entity linking or named entity disambiguation. In this task, specific surface forms are linked to entities appearing in the Web data. Often entity linking refers to restricted knowledge graphs such as DBpedia [1], YAGO [19]. A linked entity can be referred to from different surface forms (e.g. *Barack Obama* can be referred as *'Obama', 'US President'* etc.).

Focused crawling, first introduced in [5] is a well studied field and has seen a wide application for web documents, especially in dataset creation and used by major search engines [6, 4, 3]. In the context of Linked Data, crawling is similar to that on the web and usually follow the links, i.e. object properties, of seed entities to find potential *candidate entities*.

Crawling for *candidate entities* in the Linked Data context is usually achieved through a breadth-first search (BFS) approach, as implemented by LDSpider [11], and can be tailored with respect to a prioritisation of crawling order and the considered distance, i.e. the maximum number of hops in the data graph.

Given the scale and dynamics of available Linked Data on the Web, more efficient *focused crawling* approaches are required which crawl and rank candidate entities for their relevance for a given *crawl intent.* The *crawl intent* usually is specified in the form of *seed entities.* Referring to the document annotation scenario, a typical seed list might be the set of entities extracted from a specific set of documents. The ranking of candidate entities can exploit two type of measures, *connectivity* or *similarity*, between the candidate and seed entities. Such measures, exploit *lexical* as well as *structural*, respectively *graph-based* similarity metrics, which usually measure a notion of relatedness or similarity. As such, relevance assessment of entities is a critical challenge of significant importance also with respect to other entity-centric tasks such as entity retrieval, semantic search or entity recommendation.

In our work, we show that the performance of entity ranking measures depends heavily on the nature of the seed list at hand, i.e., the coherence or crawl intent (see Section 2.3.1). We investigate these observations by comparing different crawl configurations which differ with respect to their maximum hop size and the chosen relevance detection metric. As one contribution, we analyse and compute the *crawl intent* of seed entities and reflect it in our crawling configuration. Here, we exploit an intuition documented in the work by Pound et al. [17], where the importance of individual entities of a composite query or seed list differs across seeds, with more specific entities usually reflecting the search intent to a higher degree. Exploiting this observation, we specifically boost entities closer to the crawl intent through a dedicated *attrition factor* as part of our candidate entity ranking. To this end, our approach consists of the following steps: (i) seed list analysis, (ii) candidate crawling, (iii) seed list-specific candidate entity ranking, which are embedded into a crawling cycle. Our experiments show better performance and efficiency of our adaptive approach for a wide range of seed lists, compared with baseline ranking methods. In addition, insights gained from the above experiments are deemed applicable also in other scenarios, such as entity retrieval or entity recommendation, where candidate entity rankings need to be computed for a given query.

2.1.2 Related Work

Focused Web Crawling. The purpose of web crawling is manifold. Two of the widely spread use cases are the construction of datasets and that of vertical search engines like Google, Yahoo! etc. We focus our literature review mainly on the first case. Crawling data of a specific topic of information need, is also known as focused crawling. The term focused crawling was first introduced by de Bra et al.[5]. The focus has been widely shifted towards topic focused crawling.
Diligenti et al. [6] propose a focused crawling approach that used context graphs. A context graph in their case represents the link hierarchy between HTML pages and the co-occurring pages that are known to have as a target a relevant page. The main advantage of such an approach against traditional breadth-first-search crawls that the model is optimized for the global relevance of pages rather than the immediate gains that are achieved from directly connected pages given a seed list. Chakrabarti et al.[4] propose a topic focused crawling approach by assessing the relevance of a HTML page to a topic and further identifying sources (or web domains) that contain relevant documents are closely connected to the seed pages. Later on, Chakrabarti et al. [3] propose an improved approach with two main supervised modules. The modules are trained on the topic taxonomy Dmoz[5] and the corresponding documents that are assigned to the topics. The first module, serves as a source for generating new crawl targets that are passed as a priority list to the second module, which later on uses the DOM (Document Object Modeling) features[6] from every proposed target crawl page and assess its final relevance for a given topic.
McCallum et al. [13] present a task oriented crawler for building domain-specific search engines. The approach crawls pages that are relevant to a specific topic and further extract

[5]http://www.dmoz.org
[6]http://www.w3schools.com/jsref/dom_obj_document.asp

information from the documents, i.e. title, date etc., which are later on used as query fields. Tang et al. [20] use external knowledge bases from the medical domain, to improve both the relevance and quality of focused crawling.

Structured Data Crawling. Meusel et al. [14] propose a novel focused crawling approach for structured data. Their approach crawls RDFa microdata embedded in HTML pages. Further, they propose a two step approach. In the first step they use an online classifier that is trained on features extracted from the URLs of the pages, and in the second step assess the relevance of a group of pages (usually from the same source). The focus in this case is that of pages containing RDFa microdata.

Linked Data Crawling. In a closely relevant work, Isele et al. introduce an open-source crawling framework for Web data [11]. LDSpider traverses the Web of Linked Data by following RDF links between entities. Crawled data is to be stored either in files or in an RDF store. LDSpider is less effective when a focused crawl is required, as depicted in later sections of this deliverable. In our previous work, we proposed a system that iteratively crawls and captures the evolution of Linked Data sets based on flexible crawl definitions [7].

Relevance Metrics. The choice of our features used to assess the relevance of an entity for a crawl have been applied in crawling on other fields. Graph-based relevance has been used in entity linking [15]. Lexical features represent common features used in the previous focused web crawling works [4, 3, 13].

The aforementioned works differ on several aspects with respect to our work. The scope of the crawl techniques relies on HTML web pages, where the nature of the data and hence the features that can be extracted from them is different. For instance, DOM features are used in [3] to perform the focused crawling. Other works like [20, 13] propose specific task oriented crawls of domain-specific search engines and medical domain crawls based on respective knowledge bases. In contrast, our work focuses solely on focused entity crawling on Linked Datasets. Another distinguishing feature from previous work is the analysis of the starting point of the crawl, specifically the seed list of entities in our case. We configure the crawl strategies dependent on the crawl intent of the seed entities.

2.2 Problem Definition and Approach Overview

In this Section, we first describe the use case of focused crawling, and provide the formal problem definition as well as a brief overview of our approach.

2.2.1 Problem Definition

Our work aims at providing a mechanism for focused crawling of Linked Data, namely entities of relevance to a given seed list consisting of entities. While this is a task of relevance for efficiently preserving a particular partition of the Linked Data cloud in general, we would like to emphasise its use in document annotation or data enrichment scenarios. Here, traditionally named entity disambiguation or entity linking techniques are deployed to annotate documents with entities from a specific knowledge graph. While such entity annotations enable structured queries to retrieve documents of relevance to specific entities, more sophisticated semantic search needs to exploit additional knowledge from reference graphs, e.g. DBpedia. While the overall Linked Data graph might contain highly relevant information about the semantics of a specific entity, identifying the paths and entities of relevance, or the *semantic neighborhood* of a given entity, is a non-trivial task.

Specifically, the challenge is to identify and crawl the most relevant entities for a given set of seed entities. We define *focused crawling* of Linked Data as follows. Given a specific seed list of entities $S = \{e_1, \ldots, e_n\}$, the aim is to crawl and rank relevant candidate entities $C = \{e'_1, \ldots, e'_n\}$. Though seeds could commonly be represented through terms which require a disambiguation step, for simplification purposes we assume that seed entities are represented by entity URIs, for instance, referring to instances within the DBpedia graph.

To illustrate the need for scalable and focused crawling approaches, note that our earlier experiments have shown that a 2-hop crawl of a given seed lists results in 38,295 entities on average. Hence, we aim for a focused approach, where relevance of entities is computed as part of the crawling process and seeds for the following hop are determined based on their relevance to the seed list.

2.2.2 Approach Overview

In order to tackle the challenges of focused crawling, we adopt the following steps: (i) seed lists analysis, (ii) breadth-first search crawl (BFS) for candidates C, (iii) seed list-specific *candidate entity ranking*. The last two steps are embedded into a crawling cycle by using the relevant entities selected from step (iii) as next hop crawling queue for step (ii). Explained in more detail in the following section, we consider an *attrition factor* of each seed entity $e_i \in S$ to reflect the *crawl intent* during the focused crawling process. This is

based on the assumption that entities within a seed list have varying importance for the *crawl intent*, and hence, their individual impact on the ideal result set differs.
The overall focused crawling framework is summarized in Algorithm 1.

Algorithm 1 Focused Crawling of Linked Data

```
Require: seed list
Ensure: Related entities
 1: function FocusedCrawling($seedlist, depth$)
 2:     $coherence \leftarrow$ CoherenceComputation($seedlist$)
 3:     $attritionFactors \leftarrow$ AttritionFactorComputation($seedlist$)
 4:     $queue \leftarrow seedlist$
 5:
 6:     for $hop = 0 \rightarrow depth$ do
 7:         $candidates \leftarrow$ BreadthFirstCrawl($queue$)
 8:         $relatedEntities \leftarrow$ RelevanceAssessment($candidates, attritionFactors$)
 9:         $queue \leftarrow relatedEntities$
10:         if $hop == depth$ then
11:             return $relatedEntities$
12:         end if
13:     end for
14: end function
```

A focused crawling *configuration* represents an implementation of Algorithm 1, dependent on specific attributes of the candidate crawling process and seed list analysis, such as *depth* for candidate crawling and *attrition factor* of seed entities in a seed list. The corresponding configurations are described with details in Section 2.5.3.

2.3 Adaptive Focused Crawling

In this section we describe the seed list analysis and the adaptive crawling approach.

2.3.1 Seed List Analysis

Here we define and describe in detail the seed list analysis, specifically on how we compute the *seed list coherence* and *attrition factor* in Algorithm 1.

2.3.2 Seed List Coherence

We assume that crawl configurations require tailoring to the *seed list coherence*. The underlying assumption is that very specific and targeted seed lists will require different

crawling and relevance computation methods than very broad and unspecific seed lists. An example of the former is, for instance, a seed list containing entities labeled with '`Empire State Building`', '`Skyscraper`', '`Louis Sullivan`', while '`John Lennon`', '`Africa`', '`Mahatma Gandhi`' would constitute a very unspecific seed list.
We introduce a score to measure *seed list coherence*, Γ, computed as the reciprocal average shortest path between any seed entity pairs (see Equation 1). The shortest path between any two seed entities is denoted by $\varphi(e_i, e_j)$ on a given knowledge graph.

$$\Gamma = \frac{n(n-1)}{2\sum\limits_{\substack{i,j\\i<j}}^{n} \varphi(e_i, e_j)} \Leftarrow \begin{array}{c c} & \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{array}{c} e_1 \\ e_2 \\ \vdots \\ e_n \end{array} & \begin{pmatrix} 0 & \varphi(e_1, e_2) & \dots & \varphi(e_1, e_n) \\ & 0 & \dots & \varphi(e_2, e_n) \\ & & & \vdots \\ & & & 0 \end{pmatrix} \end{array} \tag{1}$$

The main intuition behind the computation of *seed list coherence* in Equation 1 is that entities with shorter paths in a given knowledge base tend to be more closely related. Hence, the average shortest path between seed entities determines the *coherence* of a seed list. In our experiments, we identify two crawl scopes based on Γ: (i) *low coherence*, and (ii) *high coherence*. We consider seed lists that follow $0 \leq \Gamma \leq \gamma$ to exhibit low coherence, while seed lists with $\Gamma > \gamma$ to exhibit high coherence, where the value of threshold was set to $\gamma = 0.5$ during our experiments, the detailed setup is described in Section 2.5.1.

2.3.3 Seed Entity Attrition Factor

As shown in our experimental evaluation, specific entities within a seed list strongly reflect the crawl intent. For example, consider the seed list {*Cologne Cathedral, Church, Architecture*}, the most specific entity *'Cologne Cathedral'*, reflects the most specific crawl intent, whereas the entities *'Church'* and *'Architecture'* provide contextual information, namely that *'Cologne Cathedral'* is a Church. Motivated by this, we assume that the relevance of specific candidate entities is dependent on the seed entity they are related to. For example, candidate entities similar to entity *'Cologne Cathedral'* will be ranked higher than entities that are similar to other seed entities.
In order to improve the *candidate entity ranking* we define the *attrition factor* $\lambda(e_i)$ for each seed entity in S. The *attrition factor* $\lambda(e_i)$ is measured as the fraction of the Katz centrality score [12] of e_i over the sum of all scores from all other entities in S in terms of the proportion of Katz centrality. The Katz score is computed w.r.t. a reference dataset (in our case DBpedia),

and entities with lower centrality score will have higher *attrition factor*, hence reflecting more strongly the *crawl intent*. The attrition factor $\lambda(e_i)$ is computed as in Equation 2.

$$\lambda(e_i) = \log \frac{C_{Katz}(e_i)}{\sum_{j=1}^{n} C_{Katz}(e_j)} \tag{2}$$

where, e_i is the seed entity, n is the number of seed entities in S, and $C_{Katz}(e_i)$ is the Katz centrality of e_i, and $e_i \in S$.

$$C_{Katz}(e_i) = \sum_{k=1}^{5} \sum_{j=1}^{n} \alpha^k (A^k)_{ji} \tag{3}$$

where, A is the adjacency matrix of entity e_i, namely the connections of e_i in the reference dataset, whereas k reflects the number of hops in the reference graph that are used to compute the adjacency matrix. The second sum in Equation 3 measures the connectivity degree between e_i and entities in the knowledge graph that are reachable within the hop k. Finally, α is an exponential penalization factor, which prefers higher connectivity degree on the earlier hops, and it takes values in the range of $\alpha \in [0, 1]$.

2.4 Candidate Crawling and Ranking

In this section, we describe the breadth-first search (BFS) crawl for candidate entities and the seed list-specific candidate entity ranking.

2.4.1 Crawl for Candidate Entities

The BFS starts with a queue $Q_i,\ i = 0...depth$ of entities from the seed list S in the first hop, and in later hops it is replaced by the candidate entities C (see Algorithm 1). With the increase of the number of hops, the number of candidate entities increases, hence the drop in crawling efficiency. The crawling *depth* defines the maximum number of hops, where we aim for a depth that provides high efficiency by means of sufficient candidates as well as short runtime.

2.4.2 Candidate Entity Ranking

Candidate entities are ranked according to their relevance to the seed entities in S. The relevance of a candidate entity to a seed list can be determined by distinguishing the following aspects.

- Pairwise relevance of a candidate entity with respect to each seed entity, i.e., $rel\{c_i, e_j\}$ where $c_i \in C$ and $e_j \in S$.
- Overall relevance of a candidate entity to the entire seed list, i.e., $rel\{c_i, S\}$ where $c_i \in C$.

Relevance of candidate entities c_i with respect to seed entities e_j is computed by using lexical distance measure, abridged with our weighting scheme consisting of the attrition factor λ, as introduced in the Section 2.3.3.

This method involves using text-based similarity metrics such as *Jaccard Similarity*, to assess the similarity between the candidate entities and seed entities. For instance, using labels and descriptions of entities, we construct term-reference vectors for each entity in the seed and candidate set, respectively S and C. This step is semi-automated as a user can configure the relevant datatype properties used to construct the term vector w_i for each seed entity e_i. The following formulas show the adapted pairwise Jaccard Similarity ($PairwiseJaccardSim^+$), where the traditional Jaccard similarity has been adopted by reflecting the crawl intent, as well as the overall candidate relevance computed through the *Entity-based Jaccard Similarity ($EJSim^+$).*

$$PairwiseEJSim^+(e_i, c_j) = \frac{1}{\lambda(e_i)} \cdot \frac{|w_i \cap w'_j|}{|w_i \cup w'_j|} \tag{4}$$

$$EJSim^+(c_j) = \frac{\sum_{i=1}^{n} PairwiseEJSim^+(e_i, c_j)}{n} \tag{5}$$

where, $\lambda(e_i)$ is the attrition factor of the seed entity e_i, w_i, w_j are the entity reference-vectors corresponding to e_i and c_j.

2.5 Experimental Setup

In the experimental setup, we introduce the datasets we consider for the crawling process and the approaches on generating the seed lists. Next, we assess the performance of the different crawling configurations, constrained for the varying parameters: (i) *seed list coherence*, (ii) *hop-size* and (iii) *crawl-candidate ranking approaches*. Consequently, we draw conclusions about the optimal *crawl parameters*. Finally, we compare our approach against existing baselines.

2.5.1 Seed Lists

Given that there are no existing datasets with *seed lists* for the focused crawling process, and since it simply represents an ad-hoc information need for a specific topic, we motivate the seed list generation based on our motivation example in Section 2.2, that is the document annotation scenario, where seed entities are directly linked to and extracted from documents in a corpus. In order to eliminate noise, potentially introduced by extracting entities from an arbitrary corpus, we directly utilise Wikipedia, where DBpedia entities can be derived directly from the hyperlinks of a source page to other Wikipedia pages, i.e. the equivalent of the Wikipedia entity http://en.wikipedia.org/wiki/Cologne_Cathedral is http://dbpedia.org/resource/Cologne_Cathedral.

We generate a sample of seed lists that are used to assess the crawling process for the parameters, such as the *seed list coherence*, *crawl intent* etc. The heuristic we rely on to generate the seed lists is to use the Wikipedia page view statistics[7]. The page views are gathered by real-world users that visit the different entity pages. In this way, our starting point for generating seed lists is the information need by the Wikipedia users, namely the top visited entity pages.

Since popular Wikipedia pages tend to be linked more consistently, we consider the top entities based on the page views from the first week of 2012, which consists of 1.6 billion page views. We have specifically chosen a period in the past to ensure well-populated and popular pages, rather than new and insufficiently annotated pages.

Next, to generate seed lists of varying *coherence*, we follow two different strategies.

1. **High-Coherence:** top–k entities from a single entity page
2. **Low-Coherence:** top–k entities extracted from entity pages that are related to the top-viewed entity page based on the Wikipedia categories

The intuition is that (1) will produce more coherent and (2) more diverse seed lists. In (1) we start with a top-viewed entity page u and extract the entities $\{u_1, u_2, ..., u_t\}$ that appear in the entity page (entities are marked by *anchors*) and rank according to their frequency. Next, we take the top–k frequent entities appearing in u, assuming that such entities are more closely related to u. We use the top–k entities as seeds.

For (2) we start again with a top viewed entity page in Wikipedia. For each entity u, we randomly obtain the category from the set of categories associated with u and retrieve all entities $\{u_1, u_2, ..., u_k\}$ in the same category. We then extract the entities $\{u'_1, u'_2, ..., u'_k\}$ appearing in the pages of $\{u_1, u_2, ..., u_k\}$ as candidates. Finally, a random sample of k entities from the accumulated candidate set are considered as low coherence seed list.

For this process, we use top–3 Wikipedia entity pages and 3 Wikipedia categories. As a result we obtain 6 seed lists with varying coherence Γ. The average coherence scores for the *high coherence* seed lists is Γ =0.55, whereas for the *low coherence* seeds we have Γ =0.34. The seed lists and the corresponding ground-truth are available for download[8].

2.5.2 Crowdsourced Ground Truth

We leverage crowdsourcing as a means to establish the ground truth in the form of a ranking of the most relevant entities within a specified n-hop neighborhood for a given seed list S. We

[7]https://dumps.wikimedia.org/other/pagecounts-raw/
[8]http://l3s.de/~fetahu/crawler_wise2015/

adopt the following steps in order to establish the ground truth for evaluation.

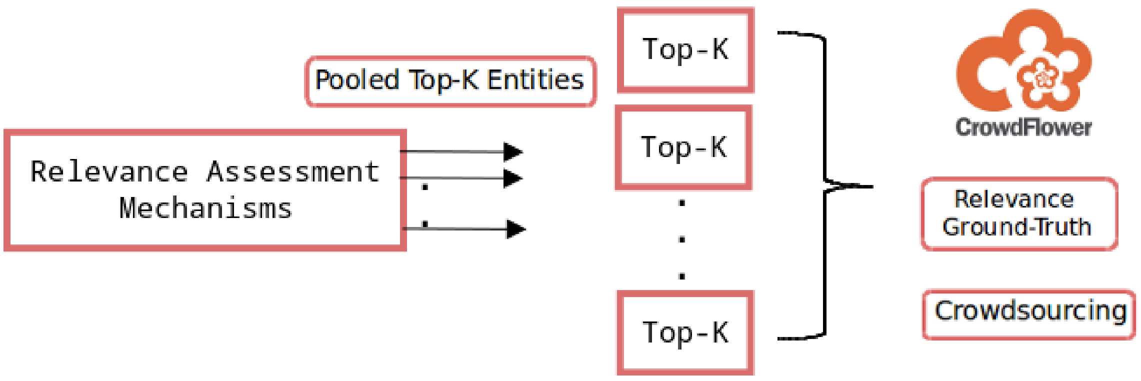


Figure 1: Generation of ground truth for evaluation of relevance assessment measures.

First, we pool the top-k entities[9], as obtained by using the different relevance assessment measures. We model the activity of assessing the relevance of an entity to the seed list as an atomic microtask that can be deployed on a crowdsourcing platform such as CrowdFlower. In order to assist the crowd workers in assessing the relevance of an entity to a given seed list, we represent the seed list with a corresponding *topic* that describes the collective meaning projected by the seed list. In order to determine an apt *topic* that can represent a seed list and thereby the crawl intent, we follow a gamified approach such as the popular ESP game proposed by Von Ahn and Dabbish [22]. Each seed list is presented to 5 different experts, who are then asked to independently tag the seed lists with representative topics. When all 5 experts reach a concensus with respect to a particular description, that tag is chosen to be the representative *topic* of the seed list. For instance, for the seed list generated from How I Met Your Mother (TV Series) Wikipedia page, experts assigned the topic *How I Met Your Mother (TV Series) - Characters and Actors.*

Next, we present each entity gathered from the neighborhood of the seed list alongside the corresponding *topic* to crowd workers and request judgments of relevance according to a 5-point Likert-scale (ranging from *Not Relevant* to *Highly Relevant*). By aggregating the judgments from the workers into relevance scores, and ranking the pooled top-k entities based on these relevance scores, we thereby generate our ground truth of relevant entities corresponding to the seed list S. Finally, we use the ground truth thus established to evaluate each of the relevance assessment methods across the top-k entities.

In order to ensure that the ground truth is reliable, we take several precautions as recommended by our previous works [10, 9] to curtail malicious activity and to avoid misinterpretations in the

[9]In this case we pooled the $Top - 500$ entities resulting from all the different configurations for each seed list.

relevance scoring.

2.5.3 Crawl Configurations and Baselines

A *crawling configuration* is defined by (a) candidate ranking approach, (b) hop size (depth) for the crawling process, and (c) seed list coherence. Here the candidate ranking considers two cases: (i) candidate relevance scoring taking into account the attrition factor λ which is denoted as $EJSim^+$, and (ii) candidate relevance scoring without the attrition factor, i.e. the baselines below. The maximum hop size *depth* considered during the crawling process is one of the most significant impact factors for runtime and NDCG score. We run experiments with *depth* in {2, 3}, the corresponding methods are denoted with a subscript indicating the depth, where, for instance, $EJSim_2$ indicates a $depth = 2$.
We consider the following baseline and state-of-the-art approaches for the focused crawling task.
EJSim. Entity Jaccard Similarity, *EJSim* computed as *EJSim+* but without considering the *attrition factor*.
NJSim. We also consider the graph-based relatedness baseline, the *Neighborhood Jaccard Similarity* which is computed as follows.

$$PairwiseNJSim(e_i, c_j) = \frac{1}{\lambda(e_i)} \cdot \sum_{k=1}^{\tau} \alpha^k \frac{|N_k(e_i) \cap N_k(c_j)|}{|N_k(e_i) \cup N_k(c_j)|} \tag{6}$$

$$NJSim(c_j) = \frac{\sum_{i=1}^{n} PairwiseNJSim(e_i, c_j)}{n} \tag{7}$$

where, $N_k(e_i)$, $N_k(c_i)$ are the sets of neighboring vertices at the k^{th} step of seed entity (e_i) and candidate entity (c_i) respectively, $\alpha \in [0, 1]$ is a real number to ensure that closer neighbors have lower weight, and τ is the maximum length of the paths considered.
PageRank. Is a widely adopted approach for ranking crawling results (documents or entities). The computation of PageRank is formalized in Equation 8.

$$PageRank(c_i) = \frac{1-d}{N} + d \sum_{c_j \in M(c_i)} \frac{PR(c_j)}{L(c_j)} \tag{8}$$

where $M(c_i)$ is the set of entities linked to candidate c_i in the entity graph, $L(c_j)$ is the number of outbound links to the entity c_j, N is the total number of entities, and d is the damping factor [2].
NB. We consider [3] as state-of-the-art approach for focused web crawling, which we adopt for our experimental setup. However, while this approach originally implements focused crawling of Web documents as opposed to Linked Data, we introduce some adaptations. The DOM tree features cannot be considered in case of Linked Data, hence, we adopt lexical features for the

NB classification. The state-of-the-art approach uses predefined topic taxonomy with example URLs as training set. In our senario, the topic, which is called crawl intent in our work, is dependent on the seed list without predefined limitations. However, our groundtruth can be used as training set for a classifier. The classification of candidate entity as relevant is performed by the function formalized in Equation 9.

$$NB(c_i) = P(y=1|c_i) = \frac{P(y)P(c_i|y)}{P(c_i)} \tag{9}$$

where $P(y=1|c_i)$ is the probability of candidate c_i belongs to class $y=1$, which in our case translates as a *'relevant'* entity for a given seed list.

2.5.4 Evaluation Metrics

To evaluate the crawling performance of the different configurations, we consider the *Normalized Discounted Cumulative Gain (NDCG)* of the ranked set of candidate entities with respect to an established ground truth. The NDCG score is computed as follows:

$$nDCG@k = \frac{DCG@k}{iDCG@k} \quad DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i}$$

where, $DCG@k$ represents the discounted cumulative gain at rank 'k', and $iDCG@k$ is the ideal $DCG@k$ computed with respect to the *ground truth* (described in the following section).
Another important factor to evaluate is the runtime performance of the different configurations. Due to the likelihood of large set of candidate entities existing for a particular seed list, runtime is a crucial factor. The runtime is simply measured in terms of the amount of time taken to complete a full-cycle of the crawling process.

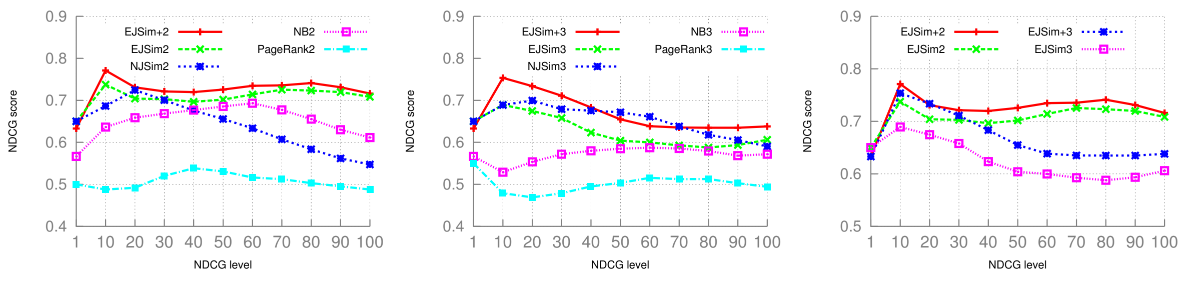
2.6 Evaluation

2.6.1 Performance of Focused Crawling Configurations

In this section we report and discuss the performance of the different focused crawling configurations. On average, for the different seed lists we crawl approximately 491,425 triples. Hence, an important aspect is how well the different crawl configurations rank the candidate entities for their relevance w.r.t the seed lists. In Table 2 we report the performance of the different crawling configurations. The values are reported for the average NDCG scores at rank 100.
Performance. Table 2 presents the average NDCG@100 score of different configurations described in Section 2.5.3 with seed lists of varying coherence. Figure 2a and 2b present the

Coherence	$EJSim_2^+$	$EJSim_2$	$NJSim_2$	NB_2	$PageRank_2$
$\Gamma < 0.5$	**0.7446**	0.7364	0.5752	0.6955	0.5346
$\Gamma \geq 0.5$	**0.6876**	0.6802	0.5193	0.5273	0.4408
	$EJSim_3^+$	$EJSim_3$	$NJSim_3$	NB_3	$PageRank_3$
$\Gamma < 0.5$	**0.6928**	0.6382	0.6608	0.6612	0.5722
$\Gamma \geq 0.5$	**0.5831**	0.5740	0.5197	0.4821	0.4166

Table 2: Average NDCG@100 for different focused crawling configurations across seed lists with varying coherence.



(a) (b) (c)

Figure 2: (a) Performance of different configurations with depth 2 measured as average NDCG across all seed lists. (b) Performance of different configurations with depth 3 measured as average NDCG across all seed lists. (c) Performance of different configurations (depth 2 and 3) measured as average NDCG across all seed lists.

NDCG score at different levels for the different configurations. Figure 2c presents a detail comparison between the performance of $EJSim$ and $EJSim^+$. It shows that the NDCG scores increased after introducing the attrition factor to EJSim. The average improvement across different NDCG levels is 1.6% on *depth* 2 and 4.3% on *depth* 3. This indicates that the *attrition factor* has a positive effect. The *coherence* of the seed list in our experimental setup does not have a significant impact on the focused crawling configuration setups.
From the results we can also observe that the $EJSim^+$ outperforms baseline methods, specifically $NJSim$ and $PageRank$. The NDCG@100 of $EJSim^+$ is 16.9% and 4.8% higher than $NJSim$ for *depth* 2 and 3 respectively. The NDCG scores at different levels (Figure 2a and 2b) also support this insight. $PageRank$ shows the weakest performance since it does not take crawl intent into consideration during candidate ranking. Furthermore, $EJSim^+$ has 7.3% and 9.8% improvement in average for *depth* 2 and 3 respectively compare to NB in our experiment. Based on the dataset, one of the reasons that NB and $EJsim^+$ have different performance while both use lexical features is that the classes within the training set are not independent from each

other. The overlap between the feature set of different classes causes negative effects on the NB classification. The reason for this overlap is that the crawl intent is based on the seed lists instead of the topic from a predefined topic taxonomy. This also reveals that our method is more robust and flexible for variety of crawl intent.

Another useful and conclusive insight is the fact that *depth* 3 has not led to performance gains with respect to *crawl depth* of 2. In fact we can observe the opposite from Figure 2c, where for $EJSim^{+}$, increasing the crawl depth beyond 2 seems to lead to weaker NDCG scores on average.

Efficiency. Since the time-intensive step is the candidate crawling, there is no considerable difference between the crawling configurations with different ranking methods. However, the average runtime across seed lists and ranking methods increased from 548 seconds of depth 2 to 1936.6 seconds of depth 3 in average while there is no significant improvement of the NDCG score. Given the significantly increased runtime when crawling beyond hop 2, a crawl depth of 2 seems to provide optimal performance and is not advisable to crawl to a higher distance.

2.7 Discussion & Limitations

Next to the aforementioned observations, we also did an analysis on more varied characteristics of seed lists. Based on the results, the following ones are worth to highlight. After the seed list size reaches a certain threshold (20 in our case), the graph-based methods seem to perform better in most cases. Meanwhile, PageRank performance seems to improve with increasing seed list size and decreasing coherence. This can be explained since Page Rank is not considering the seed list at all and the performance gap to the superior, seed list-based configurations is decreasing the more generic the seed lists become.

We evaluate the result of experiments based on a ground truth as described in Section 2.5.2. During the preliminary analysis of the dataset, we observe that the overall NDCG score for results of low coherence seed lists seems significantly higher than those of high coherence seed lists. This is due to the fact that high coherence seed lists have a more specific crawl intent, leading to narrow and often small result sets, and hence also a limited ground truth, while the low coherence lists have a much broader crawl intent as well as relevant entity set. This is reflected in our ground truth: the average number of entities labeled as related (score$\geq$ 3 and beyond) in our ground truth is 208 for low coherence seed list, and 145 for high coherence seed lists. Meanwhile, the narrow search intent also causes more disagreement among crowdsourcing workers for generating the ground truth, which makes the results for high coherence seed lists less consensual.

Another difficulty faced when evaluating the crawling task is the highly heterogeneous and varied nature of the possible result sets, originating from a highly heterogeneous Linked Data graph. While certain seed lists and crawl intents are well reflected in the Web of Linked Data, others are only sparsely represented, leading to highly varying result sets, where to some extent the availability of matching data, or the lack thereof, can have a significant impact on the measured NDCG scores. This problem is elevated due to the fact that crawl results rely on links, where the uneven distribution of links across entity types and partitions of the LD cloud leads to result sets of varying size and quality.

2.8 Lessons Learned and Future Directions

In our work, we have presented an adaptive focused crawling method which takes into account characteristic features of particular seed lists to configure the relevance assessment metric. Configurations are automatically computed based on experimentally derived heuristics and thresholds, for crawl *depth* and relevance assessment method. Our experimental results support the underlying assumption that the choice of parameters is strongly dependent on the crawl intent, which we dynamically reflect through a dedicated *attrition factor*. Our results demonstrate that the attrition factor has a positive impact on the performance, where performance can be improved significantly and all baselines are outperformed across different cases, i.e. seed lists.
Additional insights from our experiments show practical value for focused Linked Data crawling methods in general. For instance, our results suggest that crawls beyond depth 2, i.e. a distance of 2 hops in the Linked Data graph, seem not to improve the crawl quality but instead, introduce noise and significantly increase the crawl runtime.
One central obstacle when evaluating methods for the focused crawling task is the lack of a sufficient ground truth and the heterogeneity of the Linked Data graph, which naturally leads to highly diverse ground truth result sets for different seed lists. This might have a negative influence on the resulting ground truth and might dilute the performance evaluation to a certain extent. However, given the scale of our experiments, the shown results provide conclusive indicators about the performance of the investigated configurations.
While our work aimed at achieving a general performance gain, specifically measured through NDCG, our ongoing and future work foresees in particular the on-the-fly optimisation towards particular performance metrics, respectively precision, recall, runtime. While the actual crawl aims may vary strongly between different crawls - in some cases, high precision may be mandatory, in others a broad crawl, i.e. high recall will be of higher priority - through further experiments we aim at identifying more specific heuristics which can lead to a more tailored adaptation step. In this context, we are also investigating the specific characteristics of graph-based relevance metrics and lexical metrics. The high divergence of result sets produced by these two different

approaches suggests that, for instance, lexical methods are better suited to retrieve *similar* entities, while graph-based ones seems better suited to retrieve otherwise *connected* entities.

3 Clustering and Entity Retrieval

Considering the scope and the size of the crawled datasets for the semantic enrichment of the IFC files, in this section, we describe the process of querying for entities within the crawled entities. This is an important part of our SDA module since it allows users to access the data through ad-hoc queries in natural language, e.g. finding information about a particular *architect* or *building*. Correspondingly, the retrieved entities are ranked according to their relevance to a given query.
For this purpose, we propose a novel approach for the *entity retrieval*. Figure 3 shows an overview of the approach. We distinguish between two main steps: (I) *entity clustering*, and (II) *entity retrieval* which makes use of the generated entity clusters.

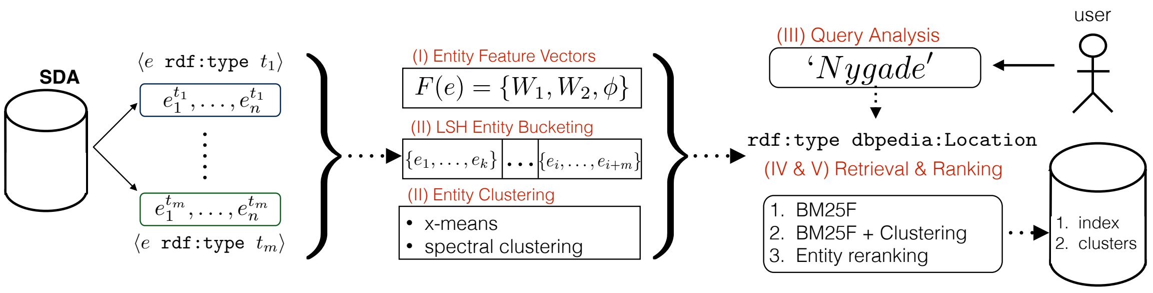

SDA
$\langle e$ rdf:type $t_1\rangle$
$e_1^{t_1}, \ldots, e_n^{t_1}$
$e_1^{t_m}, \ldots, e_n^{t_m}$
$\langle e$ rdf:type $t_m\rangle$
(I) Entity Feature Vectors
$F(e) = \{W_1, W_2, \phi\}$
(II) LSH Entity Bucketing
$\{e_1, \ldots, e_k\}$ ... $\{e_i, \ldots, e_{i+m}\}$
(II) Entity Clustering
x-means
spectral clustering
user
(III) Query Analysis
'Nygade'
rdf:type dbpedia:Location
(IV & V) Retrieval & Ranking
1. BM25F
2. BM25F + Clustering
3. Entity reranking
1. index
2. clusters

Figure 3: Overview of the *entity retrieval* approach.

Please note that the approaches in the subsequent sections are inspired from our publication in [8], for a detailed evaluation results we refer to the original publication.

3.1 Entity Clustering

In this section, we describe the offline pre-processing to cluster entities and remedy the sparsity of explicit entity links. In recent works [21] it was shown that explicit entity linking statements, specifically triples of the form $\langle e, p, e' \rangle$ where the predicate $p \in \{$`owl:sameAs`, `skos:related`, `dbp:wikiPageExternalLink`, `dbp:wikiPageDisambiguates`, `dbp:synonym`$\}$ can further improve the entity retrieval process when considering baseline approaches like the BM25F[10].
However, in Figure 4 we show that usually these linking statements are sparse in most Linked Datasets, a source from which we crawl and enrich our IFC files. Nonetheless, missing links between entities can be partially remedied by computing their pair-wise similarity, thereby complementing statements like `owl:sameAs` or `skos:related`. Given the semi-structured nature of RDF data, graph-based and lexical features can be exploited for similarity computation.

[10] https://en.wikipedia.org/wiki/Okapi_BM25

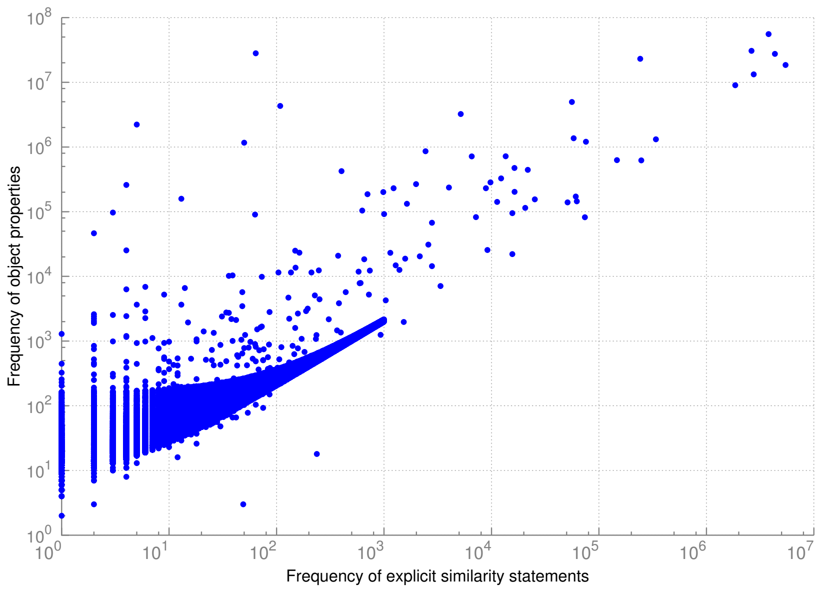


Figure 4: Number of explicit similarity statements in contrast to the frequency of object property statements overall, shown for all data graphs.

Particularly, lexical features derived from literals provided by predicates such as *rdfs:label* or *rdfs:description* are prevalent in LOD.

3.1.1 Entity Feature Vectors

Entity similarity is measured based on a set of structural and lexical features, denoted by the *entity feature vector* $F(e)$. The features for clustering are described below.

Lexical Features: We consider a weighted set of *unigrams* and *bigrams* for an entity e, by extracting all textual literals used to describe e denoted as $\mathbf{W}_1(e)$ and $\mathbf{W}_2(e)$. The weights are computed using the standard *tf–idf* metric. Lexical features represent core features when considering the entity retrieval task, more so for the clustering process. A high lexical similarity between an entity pair is a good indicator for expanding the result set from the corresponding cluster space.

Structural Features: The feature set $\phi(e)$ considers the set of all IRIs which are the subject of predicates describing e. Such an IRI represents the identifier of another entity e' which is directly related to e. The range of values for the structural features is $\phi(o, e) \to [0, 1]$, i.e., to indicate if a object value is present in e. Similarity measured by the structural features follows the principle of *SimRank*.

Feature Space: To reduce the feature space, we filter out items from the lexical and structural features that occur with low frequency across entities and presumably, have a very low impact on the clustering process due to their scanty occurrence.

3.1.2 Entity Bucketing & Clustering

Entity Bucketing. In this step we *bucket* entities of a given *entity type* by computing their *MinHash* signature, which is used thereafter by the LSH algorithm [18]. This step is necessary as the number of entities is very large. In this way we reduce the number of pair-wise comparisons for the entity clustering, and limit it to only the set of entities within a bucket. Depending on the *clustering algorithm*, the impact of bucketing on the clustering scalability varies. Since the LSH algorithm itself has linear complexity, bucketing entities presents a scalable approach considering the size of datasets in our experimental evaluation. A detailed analysis is presented in Section 2.6.

Entity Clustering. Based on the computed feature vectors, we perform entity clustering for the individual entity types and the computed LSH buckets. Taking into account scalability aspects of such a clustering process we consider mainly two clustering approaches: (i) *X–means* and (ii) *Spectral Clustering*. In both approaches we use Euclidean distance as the similarity metric. The dimensions of the Euclidean distance are the feature items in $F(\cdot)$. The similarity metric is formally defined in Equation 10.

$$d(e, e') = \sqrt{\sum (\mathbf{F}(e) - \mathbf{F}(e'))^2} \tag{10}$$

where the sum aggregates over the union of feature items from $\mathbf{F}(e), \mathbf{F}(e')$. The outcome of this process is a set of clusters $\mathbf{C} = \{C_1, \ldots, C_n\}$. The clustering process represents a core part of our approach from which we expand the entity results set for a given query, beyond the entities that are retrieved by a baseline as a starting point.

X–means To cluster entities bucketed together through the LSH algorithm and of specific entity types, we adopt an extended version of *k-means* clustering, presented by Pelleg et al. which estimates the number of clusters efficiently [16]. *X–means* overcomes two major drawbacks of the standard *k-means* clustering algorithm; (i) computational scalability, and (ii) the requirement to provide the number of clusters k beforehand. It extends the *k–means* algorithm, such that a user only specifies a range $[K_{min}, K_{max}]$ in which the number of clusters, K, may reasonably lie in. The bounds for K in our case are set to $[2, 50]$ clusters.

Spectral Clustering In order to proceed with the *spectral clustering* process, we first construct the adjacency matrix $\mathbf{A}$. The adjacency matrix corresponds to the similarity between entity pairs $d(e, e')$ of a given entity type and bucket. Next, from $\mathbf{A}$ we compute the unnormalised graph Laplacian [23] as defined in Equation 11:

$$\mathbf{L} = diag(\mathbf{A}) - \mathbf{A} \tag{11}$$

where, $diag(\mathbf{A})$ corresponds to the diagonal matrix, i.e., $diag(\mathbf{A})_{i,i} = \mathbf{A}_{i,j}$ for $i = j$.
From matrix $\mathbf{L}$ we are particularly interested in specific properties, which we use for *clustering* and which are extracted from the *eigenvectors* and *eigenvalues* by performing a singular value decomposition on $\mathbf{L}$. The eigenvectors correspond to a square matrix $n \times n$, where each row represents the projected entity into a n-dimensional space. Eigenvectors are later used to cluster entities using standard *k–means* algorithm.
However, an important aspect that has impact on the clustering accuracy, is the number of dimensions considered for the *k–means* and the k itself. We adopt a heuristic proposed in [23]. The number of dimensions that are used in the clustering step corresponds to the first spike in the eigenvalue distribution. In addition, this heuristic is also used to determine the number k for the clustering step.

3.2 Entity Retrieval

In this section, we describe the process of entity retrieval. For this task we propose a novel retrieval approach that builds upon baseline entity retrieval approach BM25F. We consider two main addition steps in this approach. First, given a user query and an initial resultset of entities retrieved by the baseline approach, we further expand the result set by making use of the pre-computed clusters in the previous section. Next, in the second step we re-rank the expanded resultset by taking account several factors such as similarity to the user query, likelihood of the particular entity type being relevant to the query etc. In details the steps are explained below.

3.2.1 Query-biased Results Expansion

Having obtained an initial result set $E_b = \{e_1, \ldots, e_k\}$ through a state of the art ER method (BM25F), the next step deals with expanding the result set for a given user query. From entities in E_b, we extract their corresponding set of clusters $\mathbf{C}$ as computed in the pre-processing stage. The result set is expanded with entities belonging to the clusters in $\mathbf{C}$. We denote the entities extracted from the clusters with E_c.
There are several precautions that need to be taken into account in this step. We define two threshold parameters for expanding the result set. The first parameter, *cluster size*, defines a threshold with respect to the number of entities belonging to a cluster. If the number is above a specific threshold, we do not take into account entities from that cluster. The underlying rationale is that clusters with a large number of entities tend to be generic and less homogeneous, i.e. they tend to be a weak indicator of similarity. The second parameter deals with the *number of entities* with which we expand the result set for a given entity cluster. The entities are considered based on their distance to the entity e_b. We experimentally validate the two

parameters in Section 2.6.

The *fit* of expanded entities $e_c \in E_c$ concerns their similarity to query q and the similarity to e_b, which serves as the starting point for the expansion of e_c. We measure the *query-biased* entity similarity in Equation 12, where the first component of the equation measures the *string* distance of e_c to q, that is $\varphi(q, e_c)$. Furthermore, this is done relative to entity e_b, such that if the e_b is more similar to q, $\varphi(q, e_b) < \varphi(q, e_c)$ the similarity score will be increased, hence, the expanded entity e_c will be penalized later on in the ranking (note that we measure distance, therefore, the lower the $sim(q, e)$ score the more similar an entity is to q). The second component represents the actual distance score $d(e_b, e_c)$.

$$sim(q, e_c) = \lambda \frac{\varphi(q, e_c)}{\varphi(q, e_b)} + (1 - \lambda) d(e_b, e_c) \tag{12}$$

We set the parameter $\lambda = 0.5$, such that entities are scored equally with respect to their match to query q and the distance between entities, based on our baseline approach. The main outcome of this step is to identify possibly relevant entities that have been missed by the scoring function of BM25F. Such entities could be suggested as relevant from the extensive clustering approaches that consider the structural and lexical similarity.

3.2.2 Query Analysis for Re-ranking

Following the motivation example in Figure 5, an important factor on the re-ranking of the result set is the *query type affinity*. It models the relevance likelihood of a given entity type t_e for a specific query type t_q. We give priority to entities that are most likely to be relevant to the the given query type t_q and are least likely to be relevant for other query types t'_q. The probability distribution is modeled empirically based on a previous dataset, BTC10. The score γ, we assign to any entity coming from the expanded result set is computed as in Equation 13.

$$\gamma(t_e, t_q) = \frac{p(t_e|t_q)}{\sum\limits_{t'_q \neq t_q} \left(1 - p(t_e|t'_q)\right)} \tag{13}$$

An addition factor we use in the re-ranking process is the *context score*. To better understand the query intent, we decompose a query q into its *named entities* and additional *contextual terms*. An example is the query $q = \{$*'harry potter movie'*$\}$ from our query set, in which case the contextual terms would be '*movie*' and the named entity '*Harry Potter*' respectively. In case of ambiguous queries, the contextual terms can further help to determine the query intent. The *context score* (see Equation 14) indicates the relevance of entity e to the contextual terms Cx of the query q. For entities with a high number of textual literals, we focus on the main literals like *labels, name* etc.

$$context(q, e) = \frac{1}{|Cx|} \sum_{c_x \in Cx} \mathbb{1}_{e \texttt{ has } c_x} \tag{14}$$

3.2.3 Top–k Ranking Model

The final step in our entity retrieval approach, re-ranks the expanded entity result set for a query q. The result set is the union of entities $\mathbf{E} = \mathbf{E}_b \cup \mathbf{E}_c$. In the case of entities retrieved through the baseline approach $e \in \mathbf{E}_b$, we simply re-use the original score, but normalize the values between $[0, 1]$. For entities from E_c we normalize the similarity score relative to the rank of entity e_b (the position of e_b in the result set) which was used to suggest e_c. This boosts entities which are the result of expanding top-ranked entities.

$$rank\_score(e) = \begin{cases} \frac{sim(q,e)}{rank(e_b)} & \text{if } e \in E_c \\ bm25f(q,e) & \text{otherwise} \end{cases} \tag{15}$$

The final ranking score $\alpha(e, t_q)$, for entity e and query type t_q assigns higher rank score in case the entity has high similarity with q and its type has high relevance likelihood of being relevant for query type t_q. Finally, depending on the query set, in case q contains contextual terms we can add $context(q, e)$ by controlling the weight of λ (in this case $\lambda = 0.5$).

$$\alpha(e, t_q) = \lambda\,(rank\_score(e) * \gamma(t_e, t_q)) + (1 - \lambda)context(q, e) \tag{16}$$

The score α is computed for all entities in $\mathbf{E}$. In this way based on observations of similar cases in previous datasets, like the BTC10 we are able to rank higher entities of certain types for specific queries.

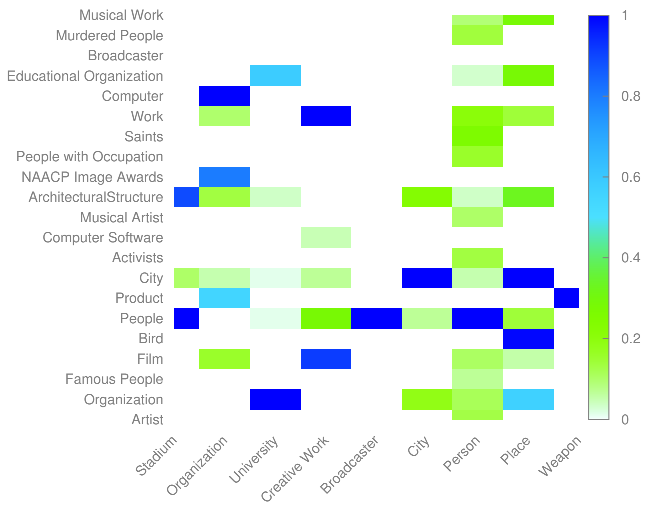


Figure 5: Query type affinity shows the query type and the corresponding entity types from the retrieved and relevant entities.

3.3 Retrieval Scenario: Indexing & Querying

In this case, as example queries can be any ad-hoc defined query consisting of an entity and other contextual terms describing it. Ideally, the entity as part of the query would have already crawled entities from the focused crawler module. Example queries could be, {*'Shreve, Lamb and Harmon'*}, an architectural firm that developed the design for the *Empire State Building*[11]. The corresponding results would be ranked according to their relevance to the particular organization *'Shreve, Lamb and Harmon'*.

One precaution here is that the index over the crawled candidate entities need to be maintained, that is, frequently update as soon as new data is crawled from the enrichment process through the focused crawler.

The advantages of such a module are manifold, (i) provides an easy access to explore the vast amount of crawled data, (ii) user-generated ad-hoc queries, and (iii) a list of entities ranked according to their relevance w.r.t the information need as conveyed by the user query.

[11] https://en.wikipedia.org/wiki/Empire_State_Building

4 Integration into the DURAARK WorkbenchUI

In this section we provide an overview of the focused crawling integration into the DURAARK WorkbenchUI.

For each method, we describe its functionality, the required parameters and the desired output. The details of the individual parameters that are required for the different methods are explained in Table 3. The DURAARK Service Platform's *Semantic Digital Archive Service* is delegating request for a semantic enrichment to the SDAS component's API. The WorkbenchUI is therefore not communicating directly with the SDAS API described here, but via the *Semantic Digital Archive Service*

4.1 Overview of Interaction within the DURAARK WorkbenchUI

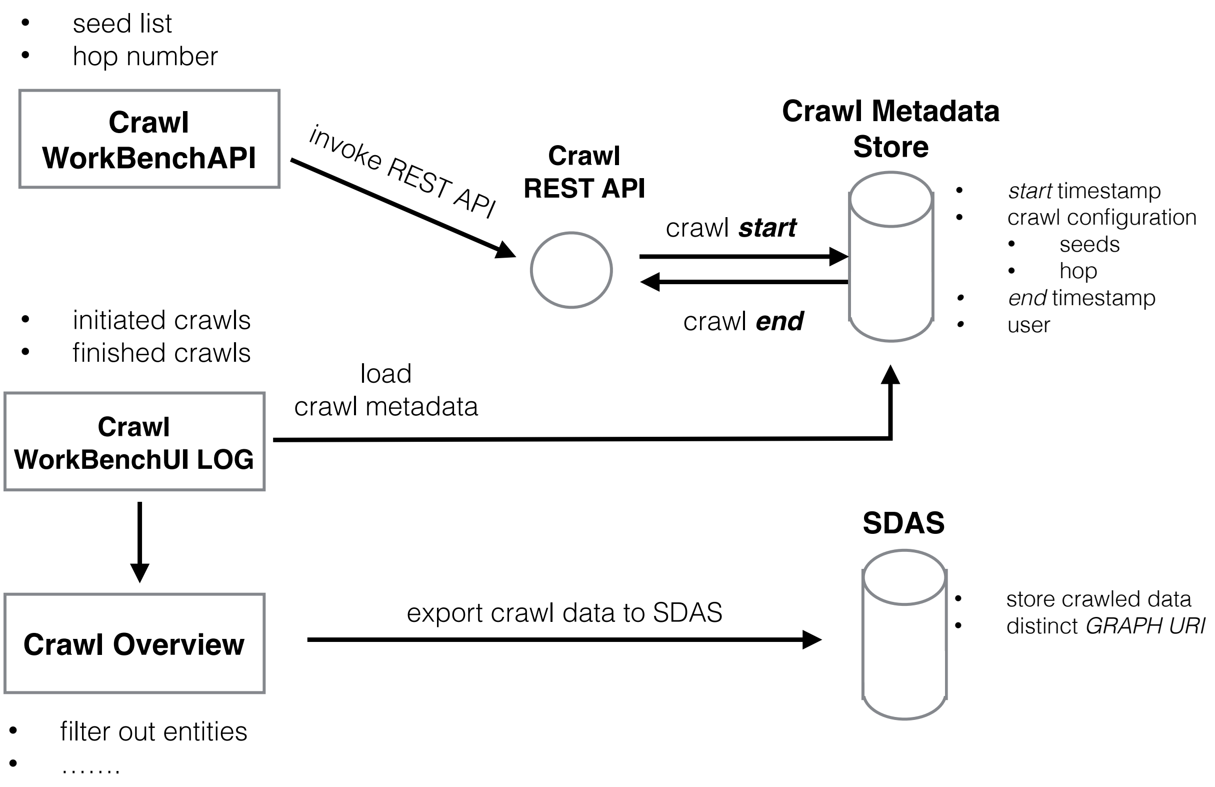
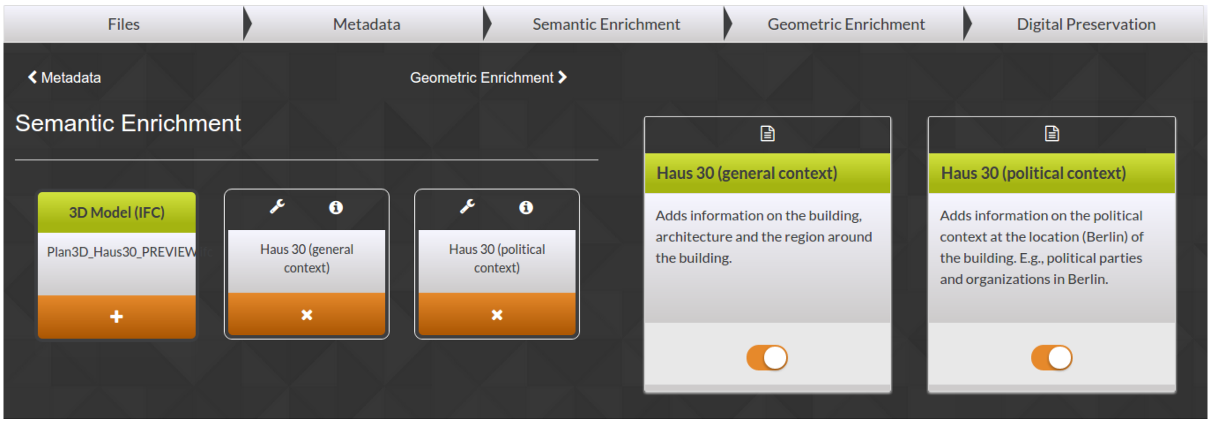


Figure 6: Semantic Digital Archive Service REST API. The diagram shows the individual steps in the focused crawling module and their corresponding interaction.

Figure 6 provides a general overview of the focused crawling module. Described in Section 2, the crawling process is initiated by first providing a seed list that encodes the *information need*. The seed list is in the form of entity URIs, usually coming from publicly available knowledge bases (i.e. DBpedia). The methods that are implemented for the focused crawling module are described in the next section. The interaction with the implemented methods is done through our DURAARK WorkbenchUI. In Figure 7 we show a screenshot of the WorkbenchUI, respectively the interaction of the WorkbenchUI with the crawler module.



Adds information on the building, architecture and the region around the building.

Adds information on the political context at the location (Berlin) of the building. E.g., political parties and organizations in Berlin.

Figure 7: Integration of the focused crawling into the DURAARK WorkbenchUI.

The individual methods are accessible under the following REST API[12] and by the specific methods names listed below. For each method, we describe its functionality, the required parameters and the desired output. The details of the individual parameters that are required for the different methods are explained in Table 3.

1. **Initiate a crawl.**
   - **Method:** *crawl*
   - **Parameters:** {*seeds, user, depth*}
   - **Output:** The crawl identification number, which is later used to perform other operations implemented in the focused crawling module.
2. **Filter crawled entity candidates.**
   - **Method:** *filterCrawlCandidates*
   - **Parameters:** {*crawl_id, crawl_filter*}
   - **Output:** Confirmation that the particular candidate entities are filtered out from the candidate entity set.
3. **Export crawl data to SDAS.**

[12] http://data.duraark.eu/services/CrawlAPI

- **Method:** *exportToSDA*
- **Parameters:** {*crawl_id*}
- **Output:** A message confirming that the crawled entity candidates are exported successfully into the SDA.

4. **Delete a crawl.**

   - **Method:** *deleteCrawl*
   - **Parameters:** {*crawl_id*}
   - **Output:** A message confirming that all information (crawled candidates and metadata regarding the crawl) from a specific crawl are deleted.

5. **Load a crawl.**

   - **Method:** *loadCrawl*
   - **Parameters:** {*crawl_id*}
   - **Output:** A ranked list of crawled candidate entities.

6. **Load the list of running crawls.**

   - **Method:** *loadAllCrawls*
   - **Parameters:** {*N/A*}
   - **Output:** The list of all *running* crawls, with the detailed crawl metadata.

7. **Load the list of all crawls.**

   - **Method:** *loadAllRegisteredCrawls*
   - **Parameters:** {*N/A*}
   - **Output:** The list of all *running/finished* crawls, with the detailed crawl metadata.

8. **Load the list of finished crawls.**

   - **Method:** *loadFinishedCrawls*
   - **Parameters:** {*N/A*}
   - **Output:** The list of all *finished* crawls, with the detailed crawl metadata.

9. **Load the list of all crawls initiated by a specific user.**

- **Method:** *loadCrawlsByUser*
- **Parameters:** {*user*}
- **Output:** The list of all crawls initiated by a specific user, with the detailed crawl metadata.

10. **Load the list of all crawls containing a specific seed entity.**

    - **Method:** *loadCrawlsBySeed*
    - **Parameters:** {*seeds*}
    - **Output:** The list of all crawls containing a specific seed entity, with the detailed crawl metadata.

parameter	methods	description
user	*crawl, loadCrawlsByUser*	The user ID or username as part of an authentication system within the DURAARK WorkbenchUI.
depth	*crawl*	The maximal depth from which we follow links into the Linked Open Data graph from the initial seed entities.
seeds	*crawl, loadCrawlsBySeed*	Seed entities coming from a specific knowledge base (e.g. DBpedia) from which we initiate a focused crawling, or in the case of *loadCrawlsBySeed* list all crawls that contain a specific seed entity.
crawl_id	*filterCrawlCandidates, deleteCrawl, exportToSDA, loadCrawl*	The crawl identification number that is used to delete, export or load the candidate entities from a specific crawl.
crawl_filter	*filterCrawlCandidates*	The filter condition that is used to delete already crawled candidate entities from a specific crawl.

Table 3: The list of parameters and their description, and the methods in which they are required.

5 DURAARK Crawls and Queries

We use our findings from the aforementioned approach towards a more targeted crawling, in order to facilitate semantic enrichment of archival data in the DURAARK context. Section 4 provides a detailed account of the interaction of the focused crawler with other components within the DUAARK workbench. In this section, we aim to provide the reader with an understanding of how the data within the SDAS can support use cases described in the Introduction of this deliverable (see UC2, UC5 and UC8).

5.1 Querying the DURAARK SDAS

We first present queries that leverage the data that is in the SDAS, enriched with data resulting from focused crawls. As presented in Section 4, crawled data is stored in the SDAS. All queries are executed on the SDAS SPARQL endpoint[13]. While the graph varies for different crawls, i.e. each crawl is stored in a separate graph, please specific the graph as `http://data.duraark.eu/crawl/`[CRAWL] where [CRAWL] is replaced with the respective crawl id, indicated in each listing. For instance, for querying crawl with id=4, please use the graph `http://data.duraark.eu/crawl/4`.

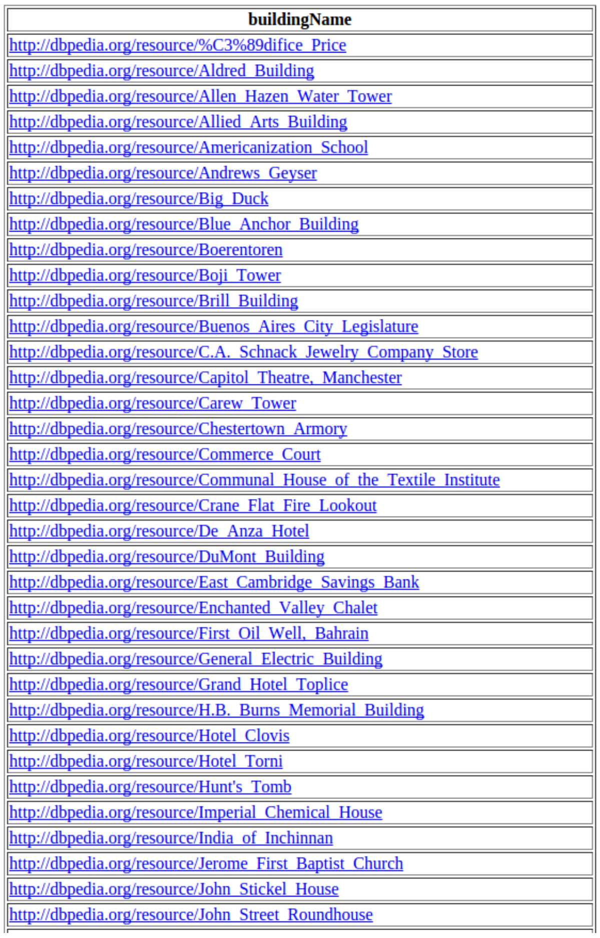
- Retrieve all buildings in the SDAS (`crawl_id`=4) that were completed in a particular year. For queries such as these, one can leverage the YAGO categories in the crawled data. Consider buildings completed in the year 1931. The query in Listing 1 retrieves buildings that were completed in the year 1931, and the Figure 8 shows an excerpt of the results.

  Listing 1: Example query to enrich SDAS building instances with relevant building entities from surrounding location (here, Manhattan).

```
SELECT ?buildingName ?type ?value FROM <http://data.duraark.eu/crawl/4>
WHERE {
    ?buildingName ?type ?value.
    FILTER regex(?value,"CompletedIn1931")
    }
```

- Retrieve all buildings of the same type from crawled data as specific instances in the SDAS. Consider the examples of ‘Berlin Catherdral’ and ‘Cologne Catherdral’. The query in Listing 2 retrieves all churches from the corresponding crawl (`crawl_id`=3).

[13] `http://data.duraark.eu/sparql`



buildingName
http://dbpedia.org/resource/%C3%89difice_Price
http://dbpedia.org/resource/Aldred_Building
http://dbpedia.org/resource/Allen_Hazen_Water_Tower
http://dbpedia.org/resource/Allied_Arts_Building
http://dbpedia.org/resource/Americanization_School
http://dbpedia.org/resource/Andrews_Geyser
http://dbpedia.org/resource/Big_Duck
http://dbpedia.org/resource/Blue_Anchor_Building
http://dbpedia.org/resource/Boerentoren
http://dbpedia.org/resource/Boji_Tower
http://dbpedia.org/resource/Brill_Building
http://dbpedia.org/resource/Buenos_Aires_City_Legislature
http://dbpedia.org/resource/C.A._Schnack_Jewelry_Company_Store
http://dbpedia.org/resource/Capitol_Theatre,_Manchester
http://dbpedia.org/resource/Carew_Tower
http://dbpedia.org/resource/Chestertown_Armory
http://dbpedia.org/resource/Commerce_Court
http://dbpedia.org/resource/Communal_House_of_the_Textile_Institute
http://dbpedia.org/resource/Crane_Flat_Fire_Lookout
http://dbpedia.org/resource/De_Anza_Hotel
http://dbpedia.org/resource/DuMont_Building
http://dbpedia.org/resource/East_Cambridge_Savings_Bank
http://dbpedia.org/resource/Enchanted_Valley_Chalet
http://dbpedia.org/resource/First_Oil_Well,_Bahrain
http://dbpedia.org/resource/General_Electric_Building
http://dbpedia.org/resource/Grand_Hotel_Toplice
http://dbpedia.org/resource/H.B._Burns_Memorial_Building
http://dbpedia.org/resource/Hotel_Clovis
http://dbpedia.org/resource/Hotel_Torni
http://dbpedia.org/resource/Hunt's_Tomb
http://dbpedia.org/resource/Imperial_Chemical_House
http://dbpedia.org/resource/India_of_Inchinnan
http://dbpedia.org/resource/Jerome_First_Baptist_Church
http://dbpedia.org/resource/John_Stickel_House
http://dbpedia.org/resource/John_Street_Roundhouse

Figure 8: Excerpt of results for buildings completed in a particular year (1931).

Listing 2: Example query to retrieve buildings of a particular type based on crawled data (here, the example of 'Church' is considered.

```
SELECT ?building FROM <http://data.duraark.eu/crawl/3>
WHERE {
    ?building ?type ?value.
    FILTER regex(?value,"Church")
}
```

- Enriching data in the SDAS with crawled data. For example, consider that building instance 'Haus 30' in the SDAS. We crawled for relevant information regarding the architect of 'Haus 30'. The results are exported to SDAS accordingly (`crawl_id`=2).

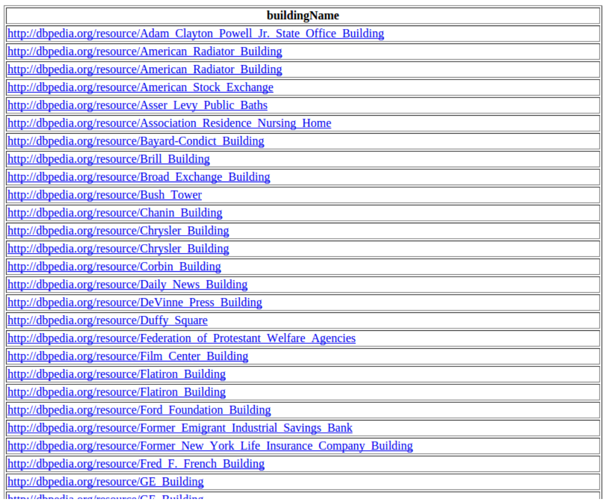
Listing 3: Example query to enrich SDAS building instance with relevant entities from crawled data.

```
SELECT ?entity ?type ?value FROM <http://data.duraark.eu/crawl/2>
WHERE {
    ?entity ?type ?value
} LIMIT 100
```

- Based on building instances and crawled data, retrieve a list of surrounding buildings in a given region. For example, consider that building instances 'Empire State Building', 'Chrysler Building' and 'Woolworth Building' in the SDAS. We crawled for relevant information regarding these skyscrapers. An excerpt of the results are presented in the Figure 9 corresponding to the query in Listing 4.

Listing 4: Example query to enrich SDAS building instance with relevant building entities from surrounding location (here, Manhattan).

```
SELECT ?buildingName
FROM <http://data.duraark.eu/crawl/sda_instances>
WHERE {
    ?buildingName ?type ?value .
    FILTER regex(?value,"Manhattan")
}
```



buildingName
http://dbpedia.org/resource/Adam_Clayton_Powell_Jr._State_Office_Building
http://dbpedia.org/resource/American_Radiator_Building
http://dbpedia.org/resource/American_Radiator_Building
http://dbpedia.org/resource/American_Stock_Exchange
http://dbpedia.org/resource/Asser_Levy_Public_Baths
http://dbpedia.org/resource/Association_Residence_Nursing_Home
http://dbpedia.org/resource/Bayard-Condict_Building
http://dbpedia.org/resource/Brill_Building
http://dbpedia.org/resource/Broad_Exchange_Building
http://dbpedia.org/resource/Bush_Tower
http://dbpedia.org/resource/Chanin_Building
http://dbpedia.org/resource/Chrysler_Building
http://dbpedia.org/resource/Chrysler_Building
http://dbpedia.org/resource/Corbin_Building
http://dbpedia.org/resource/Daily_News_Building
http://dbpedia.org/resource/DeVinne_Press_Building
http://dbpedia.org/resource/Duffy_Square
http://dbpedia.org/resource/Federation_of_Protestant_Welfare_Agencies
http://dbpedia.org/resource/Film_Center_Building
http://dbpedia.org/resource/Flatiron_Building
http://dbpedia.org/resource/Flatiron_Building
http://dbpedia.org/resource/Ford_Foundation_Building
http://dbpedia.org/resource/Former_Emigrant_Industrial_Savings_Bank
http://dbpedia.org/resource/Former_New_York_Life_Insurance_Company_Building
http://dbpedia.org/resource/Fred_F._French_Building
http://dbpedia.org/resource/GE_Building

Figure 9: Excerpt of results for buildings in Manhattan based on the crawled data relevant to skyscrapers.

In addition to exploiting crawled data in the SDAS with the use of external datasets, we can also address a wide range of queries that support additional requirements by merely leveraging the DURAARK *buildm* schema. A few examples are provided below, with respect to the Graph IRI : http://data.duraark.eu/crawl/sda_instances in the SDAS.

- Retrieving all historic buildings in a given locality (for example, Berlin). Listing 5 presents the corresponding query.

Listing 5: Retrieving buildings in the SDAS from a given locality.

```
SELECT ?s ?p ?o FROM <http://data.duraark.eu/crawl/sda_instances>
WHERE {
    ?s <http://data.duraark.eu/vocab/buildm/completionDate> ?date.
    ?s <http://data.duraark.eu/vocab/buildm/addressLocality> "Berlin".
    ?s ?p ?o
}
```

- Retrieving all buildings in the SDAS which have a particular architectural style (for example, Art Deco). Listing 6 presents the corresponding query.

Listing 6: Retrieving buildings in the SDAS based on a given architectural style.

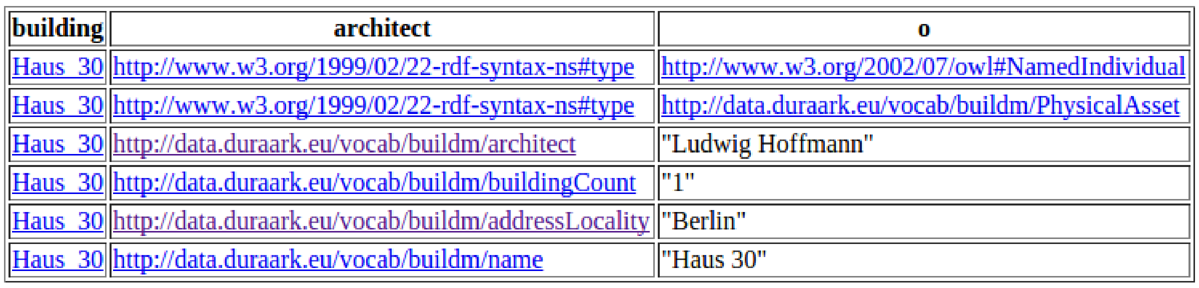
```
SELECT ?building FROM <http://data.duraark.eu/crawl/sda_instances>
WHERE {
    ?building <http://data.duraark.eu/vocab/buildm/architectureStyle> "Art
        Deco"
} LIMIT 100
```

- Retrieving details regarding all buildings in the SDAS which were designed by a particular architect (for example, Ludwig Hoffmann). Listing 7 presents the corresponding query. See Figure 10 for an excerpt of the results.

Listing 7: Retrieving data from the SDAS corresponding to a particular architect.

```
SELECT ?building ?architect ?o FROM
    <http://data.duraark.eu/crawl/sda_instances> WHERE {
    ?building <http://data.duraark.eu/vocab/buildm/architect> "Ludwig
        Hoffmann".
    ?building ?architect ?o
}
```

- Querying with multiple filters such as in the following: Retrieving number of floors in all skyscrapers in the SDAS which are located in Manhattan. Listing 8 presents the corresponding query.



building	architect	o
Haus 30	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#NamedIndividual
Haus 30	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://data.duraark.eu/vocab/buildm/PhysicalAsset
Haus 30	http://data.duraark.eu/vocab/buildm/architect	"Ludwig Hoffmann"
Haus 30	http://data.duraark.eu/vocab/buildm/buildingCount	"1"
Haus 30	http://data.duraark.eu/vocab/buildm/addressLocality	"Berlin"
Haus 30	http://data.duraark.eu/vocab/buildm/name	"Haus 30"

Figure 10: Excerpt of results from the query presented in Listing 7.

Listing 8: Retrieving number of floors in a given set of buildings in the SDAS, which are located in a given place.

```
SELECT ?building ?p ?floors FROM <http://data.duraark.eu/crawl/sda_instances>
WHERE {
    ?building <http://data.duraark.eu/vocab/buildm/addressLocality>
        "Manhattan".
    ?building <http://data.duraark.eu/vocab/buildm/floorCount> ?o.
    ?building ?p ?floors
}
```

The data in the SDAS can thereby be used to satisfy a variety of potential user needs.
For more queries that leverage the crawled data and address use cases presented earlier in the deliverable, please refer to this URL: http://data-observatory.org/sdas/.

5.2 Query Federation involving SDAS and external endpoints

Next, we present an example that entails the potential use cases that are supported by harnessing data from the SDAS together with data, or background knowledge, from external data sources, such as DBpedia or Geonames. While data in the SDAS strives to be self-contained by running crawls for data of relevance in user-driven queries, there might be use cases where additional data from the Web has to be used as background knowledge.

- Enriching SDAS data with location related information from external datasets such as 'Geonames'. Consider the following query in Listing 9 that enables us to enrich a resource situated in Stanley, Idaho with the geonames artifact as shown in the Figure 11.[14]

Listing 9: Example query to enrich SDAS resource with locations from Geonames.

[14]Note: the following queries involve query federation including remote endpoints which might have temporary downtimes.

```
SELECT ?building ?geoloc FROM <http://data.duraark.eu/test_graph> WHERE {
  {?building <http://data.duraark.eu/vocab/buildm/addressLocality> "Stanley".
  ?building ?p ?geoloc. BIND(?building AS ?location)}
UNION
{ SERVICE <http://dbpedia.org/sparql> {?location ?p ?geoloc}}. FILTER
    (regex(?p, "locatedIn"))
}
```

6 Technical Decisions and Risks

In this section, we are discussing technical decisions taken during the development of the prototypes described in this deliverable and associated risks and contingency actions.

6.1 Technical Decisions

While implementing the focused crawler and retrieval approach, a number of technical decisions were taken aimed at realising both components in their most efficient form.

6.1.1 Linked Data for Semantic Enrichment

We focused on harnessing data from the LOD cloud, since it is a rich source of structured information corresponding to various contexts relevant to DURAARK. This has been elucidated in the deliverables D3.1 through D3.4 and is inline with the general approach and earlier strategic decision, to focus on Linked Data and associated W3C standards such as RDF and SPARQL. The structured nature of linked data, the ease of integrating LD into a structured knowledge graph and its abundance were the underlying reasons behind this choice.

6.1.2 Need for a Targeted Crawler

Following experiences with the previous DURAARK crawler, we decided to implement a more targeted focused crawler due to the following reasons.

- To enable the scalability of crawling processes and the emerging enrichment: earlier versions of the crawler were aimed at retrieving entire datasets (such as DBpedia). Given the potential scale of datasets, crawling and archiving entire datasets repeatedly will accumulate large amounts of data, which are hard to manage, that is store or query in the longer term. A focused crawler on the other hand is geared towards retrieving only very small subsets of the data of crucial relevance for the project.
- To improve the accuracy of the enrichment data: given the unfocused approach of the earlier crawler, large amounts of potentially irrelevant data were stored within the SDAS, making the filtering and retrieval challenging. A focused crawler by definition populates the SDAS with potentially relevant data only and ensures the high relevance of data in the SDAS.
- To consequently optimize use of storage in the SDAS component: given the limited storage and processing capacity, efficient data crawling, relevance detection and archival

techniques are required, which limit the amount of considered data already during the crawling process, as achieved by the focused crawler presented in this deliverable.

While these reasons argue for the focused crawler implementation as implemented in this deliverable, it has to be noted that the limited amount of data and the reliance on relevance detection algorithms include the risk of potentially important data being unconsidered. This could be due to poorly formed seed lists as well as poor crawler performance. Regarding the former, we refer the reader to the evaluation results of the crawler performance presented in[24].

6.1.3 Crawl Depth

We limit the crawls to a maximum hop size of '2' due to several prior experiments that tested the optimal crawl depths, as presented in[24]. We note that crawls that reach beyond 2 hops not only take considerably longer and drastically increase the amount of involved data, but also result in inferior quality (i.e. precision) of resulting candidate entities.

6.1.4 Integration of Crawler and Entity Retrieval

We note that due to the given scale of data in the SDAS, the entity retrieval scenario presented in this deliverable is not an immediate need. While it is an important necessity in a setting that deals with large amounts of data, we envision its use in the future, during live usage of the DURAARK infrastructure involving large amounts of data. Hence, recent work focused on integrating the crawler into the DURAARK WorkbenchUI. By leveraging the data in the SDAS, and the use of SPARQL and faceted search as showcased in this deliverable, we showed that a user can benefit greatly through the current integration and retrieval methods. Further evaluation of the crawler and retrieval methods will be presented in D7.4.

6.2 Risks

During the work on this deliverable, a number of technical risks emerged, which are described below together with corresponding contingency actions.

Risk Description The use of SPARQL endpoints is replaced by other standards and future versions of Linked Data are presented differently

Risk Assessment .

Impact High

Probability Low

Description Even though they differ in implementation details, SPARQL endpoints will very likely remain to play a role in the future of linked data. Additional layers such as security etc. might be added on top which would require adaptions of the prototypical tools described here.

Contingency Solution The organisations of the DURAARK consortium are closely following the developments of the Semantic Web and Linked Data communities. If severe modifications of elemental building blocks such as SPARQL endpoints are being introduced into the overall LD approaches, conceptual and technical migration paths will very likely be developed along side in many other research initiatives and products.

Risk Description The focused crawler does not catch all relevant data due to poorly formed seed lists or a poor performance of the relevance detection mechanisms.

Risk Assessment .

Impact Medium

Probability Medium

Description High quality retrieval results rely on the high quality of the queries, that is seed lists, and the performance of the relevance detection mechanism. Regarding the latter, we refer the reader to our promising performance results in[24]. Regarding the seed list generation, further experiments with real users will be conducted as part of WP7/D7.4, providing further insights into the quality of user-defined seed lists.

Contingency Solution The WorkbenchUI aids users when providing seed lists by adopting so-called seed list templates. These represent predefined seeds for specific contextual information, which can be combined with specific buildings or locations. Based on the evaluation results, these templates can be constrained further, to relieve users from manual effort as much as possible thereby improving the quality of seed lists.

Risk Description Data in the Linked Data graph is insufficient, i.e. too static or outdated.

Risk Assessment .

Impact Medium

Probability Medium

Description While the LD graph contains a number of highly relevant datasets, a large number of datasets is outdated or not related to the DURAARK context, limiting the amount of potentially relevant datasets and data.

Contingency Solution While the quantity and quality of publicly available LD is limited, additional structured Web data is available in the form of embedded annotations, such as microdata. The take-up and scale of such data has dramatically increased throughout the last years, with current estimations claiming 25% of all Web documents containing structured embedded annotations already. Current research is investigating ways to exploit this data as a potential source for SDA enrichments, potentially complementing or substituting the LD crawler.

Risk Description Data in the LD cloud is not dereferencable.

Risk Assessment .

Impact Medium

Probability High

Description Recent studies document that LD endpoints and dereferencing interfaces show significant downtimes and limited availability, which hinders the focused crawling approach, essentially based on dereferencing RDF resources on the Web.

Contingency Solution Similar to the aforementioned risk, structured embedded annotations of Web documents will be exploited as complementary information source for enrichment of the SDA.

7 Software Licenses

The following table gives an overview of the software licences generated and used for the web services and UI modules implementation:

IPR Type	**IP used or generated**	**Software name**	**License**	**Information**
software	used	Apache Tomcat Web Server	Apache License, Version 2.0	http://tomcat.apache.org/legal.html
software	generated	Clustering & Retrieval	GNU General Public License	https://github.com/bfetahu/entity_clustering
software	used	LDSpider	GNU Lesser GPL	https://code.google.com/p/ldspider
software	generated	Focused Crawler	GNU Lesser GPL	https://github.com/bfetahu/focused_crawler
API	generated	SDAS	Creative Commons CC0 1.0 Universal Public Domain Dedication	http://data.duraark.eu/services/CrawlAPI

We have ensured that the licenses generated and used within the scope of D3.6 follow the DURAARK project's IPR strategy, i.e., we have made all components freely available for public use. This is reflected by the licenses used, as shown in Table ??.

8 Conclusions and Impact

In this deliverable, we have described the second version of the interlinking and clustering prototype (D3.4), where significant improvements have been designed and implemented. Specifically, we introduced a focused crawler for linked data, which replaces the previously developed crawling environment with a more targeted and hence scalable approach. Crawls are either based on (a) manually defined seed lists, for instance, to retrieve relevant LD subgraphs about the geographic, historical or infrastructural context of buildings and their model or (b) automatically extracted seeds, directly derived from existing BuildM instances. Based on experimentally tested crawl configurations, we introduce an efficient means to crawl linked data of relevance to the specific instances in the SDA. The focused crawler is exposed as DURAARK micro service (D2.5) and already integrated into the DURAARK WorkbenchUI and workflows. In addition, we introduce an entity retrieval approach which extends existing state of the art methods and provides improved retrieval performance, when applied to large-scale data.
The previously developed SDO (D3.2) provides the metadata (profiles) about available datasets to be crawled and as such, facilitates the detection of targeted entry points into the LOD graph for specific seed lists. While this functionality is deemed essential for scenarios where a wide range of datasets is meant to be archived/crawled, the narrow DURAARK domain (architecture) limits the potentially relevant datasets to a reasonable amount. To this end, SDO data is not used dynamically for looking up datasets for each crawling actions, but instead, it provides a sound basis for informed decisions when pre-configuring the focused crawler.
This work had an impact on DURAARK activities in a number of ways:

- WP3: providing some of the essential building blocks for populating and searching the SDA
- WP2: supporting the archival and retrieval workflows (input/output) through dedicated methods for enriching (focused crawler) and retrieving (entity retrieval/clustering) data in the SDA.
- WP6: supporting preservation of external background knowledge through targeted and scalable crawling methods. Preservation activities and actions related to the SDA data are currently under development in WP6.
- WP7: providing functionalities which enhance use cases and user experience and directly satisfy user requirements and stakeholder queries as identified in WP7.

As part of future work, the discussed and introduced components will be evaluated and tested as part of WP7 evaluation activities (upcoming D7.4). A deeper integration of components is envisaged as well as more large-scale deployment and validation based on increasing quantities of archived data.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *International Semantic Web Conference (ISWC)*, pages 722–735, 2007.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[3] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 148–159, New York, NY, USA, 2002. ACM.

[4] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640, 1999.

[5] Paul De Bra, Geert-Jan Houben, Yoram Kornatzky, and Renier Post. Information retrieval in distributed hypertexts. In *RIAO*, pages 481–493, 1994.

[6] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C Lee Giles, Marco Gori, et al. Focused crawling using context graphs. In *VLDB*, pages 527–534, 2000.

[7] Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze. Crawl me maybe: Iterative linked dataset preservation. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014.*, pages 433–436, 2014.

[8] Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze. Improving entity retrieval on structured data. In *The 14th International Semantic Web Conference, ISWC*, 2015.

[9] Ujwal Gadiraju, Ricardo Kawase, and Stefan Dietze. A taxonomy of microtasks on the web. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 218–223. ACM, 2014.

[10] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. Understanding malicious behaviour in crowdsourcing platforms: The case of online surveys. In *Proceedings of CHI'15*, 2015.

[11] Robert Isele, Jürgen Umbrich, Christian Bizer, and Andreas Harth. Ldspider: An open-source crawling framework for the web of linked data. In *9th International Semantic Web Conference (ISWC2010)*. Citeseer, 2010.

[12] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, VOL. 18, NO. 1:39– 43, 1953.

[13] Andrew McCallumzy, Kamal Nigamy, Jason Renniey, and Kristie Seymorey. Building domain-specific search engines with machine learning techniques. 1999.

[14] Robert Meusel, Peter Mika, and Roi Blanco. Focused crawling for structured data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1039–1048, 2014.

[15] Bernardo Pereira Nunes, Stefan Dietze, Marco Antonio Casanova, Ricardo Kawase, Besnik Fetahu, and Wolfgang Nejdl. Combining a co-occurrence-based and a semantic measure for entity linking. In *The Semantic Web: Semantics and Big Data*, pages 548–562. Springer, 2013.

[16] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[17] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 771–780. ACM, 2010.

[18] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[19] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[20] Thanh Tin Tang, David Hawking, Nick Craswell, and Kathy Griffiths. Focused crawling for both topical relevance and quality of medical information.

[21] Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *The 35th International ACM SIGIR, Portland, OR, USA, August 12-16, 2012*, pages 125–134, 2012.

[22] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[23] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[24] Ran Yu, Ujwal Gadiraju, Besnik Fetahu, and Stefan Dietze. Adaptive focused crawling of linked data. In *The 16th Web Information System Engineering*, 2015.