# LOTUS: Linked Open Text UnleaShed

Filip Ilievski, Wouter Beek, Marieke van Erp,
Laurens Rietveld and Stefan Schlobach

The Network Institute
VU University Amsterdam
{f.ilievski,w.g.j.beek,marieke.van.erp,
l.j.rietveld,k.s.schlobach}@vu.nl

**Abstract.** It is difficult to find resources on the Semantic Web today, in particular if one wants to search for resources based on natural language keywords and across multiple datasets. In this paper, we present LOTUS: Linked Open Text UnleaShed, a full-text lookup index over a huge Linked Open Data collection. We detail LOTUS' approach, its implementation, its coverage, and demonstrate the ease with which it allows the LOD cloud to be queried in different domain-specific scenarios.

**Keywords:** Findability, Text Indexing, Semantic Search, Big Data

## 1 Introduction

Linked Open Data sources, such as those found in the Linked Open Data Cloud and the LOD Laundromat, hold a promise to many users for relatively easy access to a wealth of information. Simultaneous querying of a large number of data sets is facilitated through services such as the LOD Laundromat[1]. Unfortunately, it has always been difficult to *find* resources on the Semantic Web. A resource is typically 'found' by memorizing its resource-denoting IRI and/or by memorizing the IRI of a SPARQL endpoint in which descriptions about that resource are disseminated. It has been especially difficult to find resources based on textual descriptions that are associated to that resource. In cases where a text index for Semantic Web data has been built, such an index has usually been restricted to a single dataset. The findability problem of the Semantic Web poses an extra challenge for people that come from outside of the Semantic Web community and wish to use semantic resources.

In this paper, we present LOTUS, a full-text lookup index over the Linked Open Data cloud in the LOD Laundromat. Text search on the LOD cloud is not new, as Sindice[1] and LOD Cache[2] show. However, to ready the LOTUS index for text analysis tasks, a completely text-centric approach is taken. The LOTUS index focuses on the string values present in RDF statements to create a fast and scalable index over the data in the LOD Laundromat[3] that can easily be

[1] http://www.sindice.com/, discontinued in 2014.
[2] http://lod.openlinksw.com/
[3] The LOD Laundromat contained 38,606,408,433 triples on 6 July 2015

queried through an API or web interface and provides an option to search for strings in particular languages through an associated language tag. LOTUS is a first step up to an accessible disambiguation system over the LOD cloud, to be used for example in NLP applications, that currently suffer from lack of coverage of resources such as Wikipedia/DBpedia[8]. Furthermore, as LOTUS provides a link between text and documents in the LOD cloud, Information Retrieval over the LOD cloud becomes an interesting option. In Section 6, we will discuss some use case scenarios. Our contributions are the following:

- A problem description of accessing textual resources on the Semantic Web today (Section 2)
- The LOTUS system (Sections 4 and 5)
- Three use cases that showcase the potential of LOTUS (Section 6)

## 2 Problem description

How do we find relevant resources on the Semantic Web today? The Semantic Web currently largely relies on two findability stategies: IRI dereferencing and SPARQL query endpoints.

**Dereference** According to Semantic Web folklore an IRI $c$ should dereference to a set of expressions $\Phi(c)$ in which that IRI appears (sometimes only in the subject position, but this is immaterial to the current argument).

Which expressions belong to $\Phi(c)$ is decided by the *authority* of $c$, i.e., the person or organization that pays for the domain that appears in $c$'s authority component. Non-authoritative expressions about $c$, denoted $\overline{\Phi(c)}$, are all those expressions in which $c$ occurs (again, possibly in the subject term position) but that are not part of $\Phi(c)$.

Findability of non-authoritative expressions occurs through an alternating sequence of IRI dereference and graph traversal operations. Even though expressions in $\overline{\Phi(c)} \cup \Phi(c)$ by definition belong to the same graph component, it may well be possible that no path from terms in $\Phi(c)$ to terms in $\overline{\Phi(c)}$ exists. Indeed, since the Semantic Web does not implement the notion of backlinks, an architectural decision it has in common with the WWW, dereferenceability is inherently unable to solve the findability problem.

In addition to its theoretical limitations, the real-world implementation of dereferenceability has proven to be both difficult and costly, since it requires a Web Server to run on the dereferenced IRI's authority. It is not uncommon for IRIs to be unavailable, either temporarily or permanently.

Since only IRIs can be dereferenced, natural language access to the Semantic Web cannot be gained at all through dereference. It is therefore not possible to find a resources based on RDF literals to which it is related. It is certainly not possible to search for resources based on keywords that only bear a close similarity to (some of the) literals to which those resources are related.

**SPARQL endpoints** The second approach towards solving the findability problem is the use of SPARQL endpoints. When compared to dereferencability-based graph traversal, SPARQL endpoints provide a far more powerful approach for finding an individual resource $c$ based on the authoritative expressions $\Phi(c)$ about it. As for the findability of non-authoritative expressions about $c$, SPARQL endpoints have largely the same problems as the dereferenceability approach.

While it is possible to evaluate a SPARQL query over multiple datasets, thereby including expressions in $\overline{\Phi(c)}$ as well, these datasets have to be included explicitly by using the `SERVICE` keyword [15]. This requires that all endpoints which disseminate expressions in $\overline{\Phi(c)}$ are know, making the findability approach somewhat circular. While there are potentially many such endpoints (depending on $c$) the list will have to be assembled anew for each new term.

In addition to the unpracticality of finding all non-authoritative endpoints that say something about $c$, there is no guarantee that all expressions in $\overline{\Phi(x)}$ are disseminated by some SPARQL endpoint. Empirical studies show that SPARQL endpoint availability is generally quite low, indicating that some SPARQL-disseminated expressions may happen to be unavailable at query time [2].

The SPARQL query language is largely oriented towards matching graph patterns and datatyped values. As such the SPARQL specification defines regular expression-based operations on the lexical expressions of literals but does not include string similarity matches or other more advanced NLP functionality.

Summarizing, findability is a problem on the Semantic Web today. The findability problem will not be solved by implementing existing approaches or standards in a better way, but requires a completely novel approach instead.

## 3 Related work

LOTUS bears much resemblance to Sindice [18], a system that allowed search on Semantic Web documents based on IRIs and keywords that appeared in those documents. Sindice crawled the network of dereferenceable IRIs and queryable SPARQL endpoints to gather data documents. The contents of each document were included in two centralized indices: one for text and one for IRIs. Sindice also semantically interpreted inverse functional relations, e.g. mapping telephone numbers onto individuals. Currently, LOTUS does not perform any type of semantic interpretation, although such functionality could be build on top of it.

There are several differences between LOTUS and Sindice. Some of these are due to the underlying LOD Laundromat architecture and some to the LOTUS system itself. Firstly, Sindice can relate IRIs and keywords to *documents* in which the former occur. LOTUS can do much more (see Figure 1): it can relate keywords, IRIs and document to each other (in all directions).

Secondly, Sindice requires data to adhere to the Linked Data principles. Specifically, it requires an IRI to either dereference or be queryable in a SPARQL endpoint. LOTUS is build on top of the LOD Laundromat which accepts any type of Linked Data, e.g. it allows data entered through Dropbox.

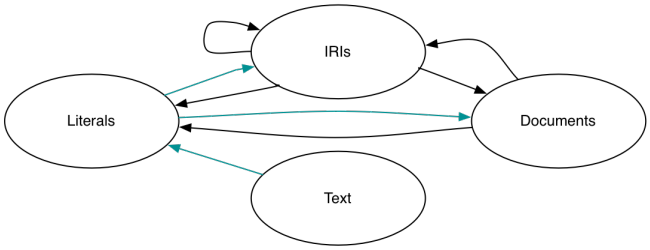


**Fig. 1.** Representation of the capabilities of LOTUS, providing the missing aquamarine link between standard Semantic Web resources and text and Literals

Thirdly, LOTUS allows incorrect datasets to be partially included due to the cleaning mechanism of the LOD Laundromat. This is an important feature since empirical observations over the LOD Laundromat collected indicate that at least 70% of crawled data documents contain bugs such as syntax errors.

Fourthly, since Sindice returns a list of online document links as a result, it relies on the availability of the original data sources. While it has this in common with search engines for the WWW, it is known that data sources on the Semantic Web have a much lower availability [6]. LOTUS returns document IRIs that can either be downloaded from their original sources or from a cleaned copy made available through the LOD Laundromat Web Service.

Fifthly, because LOTUS operates on a much larger scale than Sindice did. Sindice allowed 30M IRIs and 50M literals to be searched while LOTUS allows 3,700M IRIs and 5,320M literals, a difference in scale between Sindice and LOTUS of more than factor 100.

Various systems exist that provide the specific functionality of finding a Semantic Web resource based on natural language text input (e.g DBpedia Spotlight[11], Babelfy[12] and NERD tools[16]). The purpose of these systems is to annotate every entity ([11], [12], [16]) or concept([12]) in natural text, by linking it to an appropriate RDF resource in a background knowledge source. A key difference between these systems and LOTUS is the choice of background knowledge sources. LOTUS performs text-to-resource matching over the whole LOD cloud, while the annotation systems are generally bound to one (or a small amount of) knowledge sources: DBpedia Spotlight links to DBpedia, NERD to DBpedia and Freebase, and Babelfy to BabelNet. The focus on a single knowledge source makes generalization over new knowledge sources problematic. Recently, [10] discussed the monopoly of several knowledge bases for evaluation and solution design of entity annotation systemsAn example problematic case are the "NIL entities": entity phrases without a corresponding resource in the chosen knowledge base (e.g. around 1/5 of the entities in the AIDA/CoNLL[5] dataset are labeled as "NIL").

# 4 LOTUS

The purpose of LOTUS is to relate unstructured (natural language text) data to structured data using RDF as the paradigm for expressing such structured data. LOTUS has access to an underlying architecture that exposes a large collection of resource-denoting terms and structured descriptions of those terms, all formulated in RDF.

## 4.1 Described resources

RDF defines a graph-based data model in which resources can be described in terms of their relations to other resources. The textual labels denoting some of these resources provides an opening to relate unstructured to structured data.

An RDF statement expresses that a certain relation holds between a pair of resources.We take a **described resource** to be any resource that is denoted by at least one term that appears in the subject position of an RDF statement.

LOTUS does not allow every resource in the Semantic Web to be found through natural language search, as some described resources are not denoted by a term that appears in the subject position of a triple whose object term is a textual label. Fortunately, many Semantic Web resources have at least one textual label linked to it and as the Semantic Web adheres to the Open World Assumption, resources that have no textual description today may receive one tomorrow, as everyone is free to add new content.

We do not know how many resources are currently described by LOTUS. This is also due to the Open World Assumption which — as in natural language — does not generally allow us to ascertain that distinct terms also denote distinct resources.

## 4.2 RDF Literals

In the context of RDF, textual labels appear mainly as part of *RDF literals*. An RDF literal is either a pair $\langle D, LEX \rangle$ or a triple $\langle \texttt{rdf:langString}, LEX, LT \rangle$ [3]. $D$ is a datatype IRI denoting a datatype. $LEX$ is a Normal Form C (NFC) Unicode string [4]. $LT$ is a language tag identifying an IANA-registered natural language per BCP 47 [14].

Semantically speaking, RDF literals denote resources, similar to the way in which IRIs denote resources. A datatype $D$ defines the collection of allowed lexical forms (**lexical space**), the collection of resources denoted by those lexical forms (**value space**), a functional mapping from the former to the latter and a non-functional mapping from the latter to the former.

We are specifically interested in literals that contain natural language text. However, not all RDF literals express – or are intended to express – natural language text. For instance, there are datatype IRIs that describe a value space of date-time points or polygons. Since we are working with RDF data we cannot rely on a whitelist of datatype IRIs in order to extract all and only natural language texts from the LOD Cloud. Firstly, there is no fixed set of datatypes

and datatype IRIs as datasets can define their own. Secondly, even if we would settle for a partial whitelist we would not be able to denote a collection of datatype IRIs that *only* denoted natural language texts. While natural language text is often found together with datatype IRI `xsd:string`, in practice we find that integers and dates are also stored under that datatype, even though custom datatypes for integers and dates exist. Because of these reasons, we filter the lexical expressions to include through a pattern regardless of the datatype IRI associated with it.

### 4.3 Offline approximate string matching

LOTUS aims to facilitate flexible traversal from natural language text to RDF literals and, from there onwards to RDF resources and documents. Therefore the string matching method is a central design decision. LOTUS performs offline approximate string matching. Approximate string matching[13] is an alternative to exact string matching, where a given pattern is matched to text while still allowing a number of errors. This operation requires a measure of string similarity. LOTUS preprocesses text and builds the data index offline. This allows the approximation model to be enriched with TF-IDF term weighting score[7], phonetic matching, synonym matching, match granularity (phrase- or term-based match).

The correct approximation model depends on the intended application: lookup for journal paper titles may work better with phrase matching and substring algorithms; lookup for street addresses may deal with many abbreviations, hence it would benefit more from fuzziness and synonym search; keyword search in abstracts would prefer terms-based search, etc. As LOTUS is intended as a general-purpose system, aiming to support various applications, we plan to gradually build up towards a rich set of string matching functions.

## 5 Implementation

The LOTUS system architecture consists of two components: Index Building (IB) procedure and Public Interface (PI). The role of the IB component is to index strings from LOD Laundromat; the role of PI is to expose the indexed data to users for querying. The two system components are executed sequentially: data is initially indexed, then the indexed data can be queried through the exposed public interface.

### 5.1 System Architecture

Indexing of data is expensive. Hence, initially we create an index over all data from LOD Laundromat through a batch loading procedure, by streaming LOD Laundromat statements (both triples and N-quads) through its query interface, Frank, to a client script. The client script parses the received RDF statements

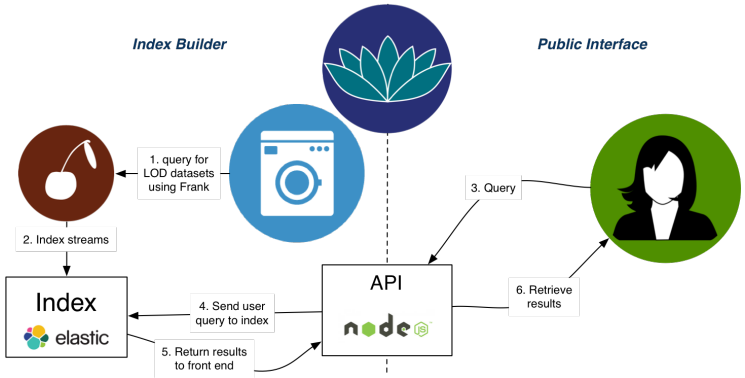


**Fig. 2.** LOTUS System Architecture

and performs a bulk indexing request in Elastic,[4] where the textual index is built. Once the initial indexing is finished, we only incrementally update the index when new data appears in LOD Laundromat, triggered by an event handler in LOD Laundromat.

We use Frank's *Frank documents* command to enumerate all LOD Laundromat data sets and download them sequentially in a stream (Step 1 in Figure 2). Following the approach described in Section 4, we consider only the statements that contain a literal as an object and use the regular expression "`/^[-\.,0-9]*$/`" to filter out statements with numbers and dates as lexical forms. The selected statements are then piped through a transformation step in which every incoming RDF statement is parsed and the resulting information is indexed in Elastic (Step 2 in Figure 2). Each Elastic entry has the following format:

```
{
  "docid": IRI,
  "langtag": STRING,
  "predicate": IRI,
  "string": STRING,
  "subject": IRI
}
```

The fields "string" and "langtag" (language tag) are preprocessed ("analyzed") by Elastic at indexing time, which allows for flexible, fuzzy lookup of these fields. The motivation behind analyzing the "string" field comes naturally, as this contains unstructured text and will rarely be queried for exact match. We also analyze the language tag field: following BCP 47 semantics[5], a language tag can easily contain subtags, for instance, country codes. In order to also retrieve the specific language tags (e.g. "en-US") when looking for general language tags (e.g. "en"), we decided to analyze the language tag field and allow for flexible

[4] `https://www.elastic.co/`
[5] `https://tools.ietf.org/html/bcp47`

matching. The remaining three fields can be looked up as exact matches, as these fields are IRIs and contain structured text following the RDF standard.

### 5.2 API

Users can search the underlying data through the API. The general query flow is described in steps 3-6 of Figure 2. The NodeJs[6] interface to the indexed information currently exposes the following four query types:

-*Q1: terms(pattern, size)*:Disjunct lookup of set of terms (supplied via the *pattern* parameter) occurring in the string field of an entry. Each hit must contain at least one of the requested terms. Hits containing more of the terms have a higher score. The best *size* hits are returned.
-*Q2: langterms(pattern, size, langtag)*:Similar as the previous call. Additionally, a *langtag* value is supplied, expressed as a preference: the hits which contain the preferred language tag will receive higher score.
-*Q3: phrase(pattern, size)*:Lookup for a phrase (supplied via the *pattern* parameter) occurring in the *string* field of an entry. Each hit must contain the phrase as a whole. The best *size* hits are returned.
-*Q4: langphrase(pattern, size, langtag)*:Similar as the previous call. Additionally, a *langtag* value is supplied, expressed as a preference: the hits which contain the preferred language tag will receive higher score.

LOTUS is also available as a web interface at `http://lotus.lodlaundromat.org/` for simple exploration of the data. The code of the API functions and data from the use cases in Section 6 are available on github.[7]

### 5.3 Indexed Data

Statistics over the indexed data are presented in Table 1. We encountered over 12 billion literals in LOD Laundromat. We filtered all numbers and date literals, summing up to 55.7% of the overall amount. The initial LOTUS index was created in 56 hours and takes 484.77 GB of disk space storage. The current index consists of 5.3 billion entries, coming from 508,244 distinct sources.[8]

There are 713 distinct terms occurring in the *langtag* field. Figure 3 presents the proportion of the 10 most frequent language tags. "en" is by far the most frequently encountered tag: 1,049,037,147 literals have been tagged with an English language tag, followed by 165,996,755 German tags and 149,507,401 French tags. Figure 3 also shows the proportion of the 10 most popular languages with respect to the overall set of 5.3 billion literals: most of the literals have no language tag assigned to them.

[6] `https://nodejs.org`
[7] `https://github.com/filipdbrsk/LOTUS\_Search/`
[8] The number of different source documents in LOTUS is lower than the overall number of sources in LOD Laundromat, as not every source document contains string literals.

**Table 1.** Statistics over the indexed data

| | |
|---|---|
| total # literals encountered | 12,018,939,378 |
| # integers and dates | 6,699,148,542 |
| # indexed entries (=# string literals) | 5,319,790,836 |
| # distinct sources | 508,244 |
| # distinct language tags terms | 713 |
| # hours to create the index | 56 |
| disk space used | 484.77 GB |

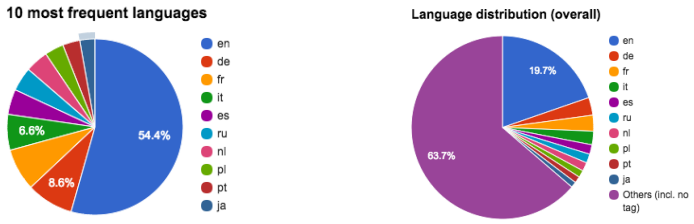


**Fig. 3.** Distribution of language tags over the LOTUS literals

## 6 Use cases

To illustrate the need for access to multiple datasets, we perform a small recall evaluation on a standard benchmark dataset, namely the CoNLL/AIDA Named entity linking benchmark dataset[5]. We also present two domain-specific use cases, namely Local monuments and Scientific journals.

For each use case scenario, we gather a set of entities and query each entity against LOTUS. We also counted the amount of entities without results and the proportion of DBpedia resources in the first 100 candidates, as a comparison to the (currently) most popular knowledge base. We then inspected a number of query results to assess their relevance to the search query. The obtained quantitative results are presented in Table 2. In the remainder of this section we detail the specifics of each use case.

### 6.1 CoNLL/AIDA

The CoNLL/AIDA dataset[5] is an extension of the CoNLL 2003 Named Entity Recognition Dataset [17] to also include links to Wikipedia entries for each entity. The CoNLL/AIDA dataset contains 7,112 'unknown' entities. We focus on these to show the impact of having access to multiple datasets. We removed the duplicate entities in each article, providing us with 5,628 entity mentions.

We suspect that the growth in DBpedia since the release of this dataset has improved recall on the named entities, but there is still a benefit of using multiple data sources. For smaller locations, such as the "Chapman Golf Club", relevant results are found in for example `http://linkedgeodata.org/About`.

**Table 2.** Use case statistics by # of queries, # queries for which no result is retrieved, # queries for which we only find resources other than DBpedia and proportion of DBpedia resources in the first 100 results per query type

| | CoNLL/AIDA | | | | Local Monuments | | | | Journals | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 |
| # queries | 5,628 | | | | 191 | | | | 231 | | | |
| # no result | 54 | 54 | 1,286 | 1,286 | 3 | 3 | 23 | 23 | 0 | X | 10 | X |
| # no DBpedia | 185 | 115 | 1,723 | 1,701 | 4 | 3 | 26 | 25 | 15 | X | 49 | X |
| % DBpedia | 64.35 | 79.11 | 69.49 | 77.68 | 67.19 | 84.92 | 70.48 | 83.23 | 22.33 | X | 24.83 | X |

Also, the fact that the different language versions of dbpedia are plugged in helps in retrieving results from localised DBpedias such as for "Ismaïl Boulahya", a Tunesian politician described in `http://fr.dbpedia.org/resource/Isma\%C3\%AFl\_Boulahya`. Some of the retrieval is hampered by typos, such as "Allan Mullally" in the newspaper text and its derived query where the cricketer "Alan Mullally" is meant.

### 6.2 Local Monuments Guided Walks

The interest in applications such as Historypin (`http://www.historypin.org`) or the Rijksmuseum API (`https://www.rijksmuseum.nl/en/api`) shows that there are interesting use cases in cultural heritage and local data. To explore the coverage of this domain in the LOD Laundromat, we created the local monuments dataset by downloading a set of guided walks from the Dutch website `http://www.wandelnet.nl`. We specifically focused on the tours created in collaboration with the Dutch National Railways as these often take walkers through cities and along historic and monumental sites. From the walks 'Amsterdam Westerborkpad' and 'Mastbos Breda', a human annotator identified 112 and 79 entities respectively. These are mostly monuments such as 'De Grote Kerk' (*The big church*) or street names such as 'Beukenweg' (*Beech street*).

We manually inspected the top 10 results on a number of queries. Here we find that the majority of the highest ranking results is still coming from DBpedia. However, when no DBpedia link is available, often a resource from the Amsterdam Museum (`http://semanticweb.cs.vu.nl/lod/am/`) or Wikidata (`http://www.wikidata.org`) is retrieved. The focus on entertainment in DBpedia is also shows here for the query 'Jan Dokter, the person who first walked the route to commemorate his family who died in WWII. 'Jan' is a very common Dutch first name, and 'Dokter' means 'doctor', which results in many results about characters in Dutch and Flemish soap operas who happen to be doctors. We intend to expand the search functionality in LOTUS towards named entity disambiguation, allowing more context to be brought into the search query to filter results better.

### 6.3 Scientific Journals

Whitelists (and blacklists) of scientific journals are used by many institutions to gauge the output of their researchers. They are also used by researchers interested

in the scientific social networks. One such list is made publicly available by the Norwegian Social Science Data Services Website (`http://www.nsd.uib.no/`). Their level 2 publishing channels contains 231 titles of journals. The majority of these titles is in English, but it also contains some German and Swedish titles barring the use of the language tag in querying.

As the queries are generally longer and contain more context-specific terms such as "journal", "transactions", "methods", and "association", the query results are generally more relevant and fewer come from DBpedia. Instead, a large part of the queries come from sources ZDB (`http://dispatch.opac.dnb.de/LNG=DU/DB=1.1/`) the 2001 UK's Research Assessment Exercise as exposed through RKB Explorer (`http://rae2001.rkbexplorer.com/`), Lobid (`http://lobid.org/`) and again Wikidata. The more generic titles, such as "Transportation" yield, as expected, more generic results.

## 7 Discussion

As mentioned Section 4, no universal string matching strategy is expected to fit all use cases. Term-based string matching is more flexible and yields more results than phrase-based matching. The right trade-off between these two strategies is application-dependent and should be further investigated.

In our use cases, we were able to find resources for the majority of the entities, but many of the results still come from generic data sets such as DBpedia. Still, the different use case show that this proportion differs per domain, opening up new perspectives and challenges for application areas, such as Named Entity Disambiguation and Information Retrieval.

Finally, LOTUS currently lacks integration of structured and unstructured data. We now allow a transition from natural language text to literals, documents and resources; but relations between the structured data are currently missing preventing the user from knowing which of the retrieved resources are identical, similar or share the same context in a certain sense.

## 8 Conclusion and Future Work

In this paper, we presented the LOTUS system, a full text look-up application over the LOD Laundromat Linked Open Data collection. We detailed the specific difficulties in accessing textual content in the LOD cloud, and demonstrated the potential of LOTUS in three small use case scenarios.

Further development of LOTUS will focus on improving performance and enriching functionality to support third-party applications, in particular for named entity disambiguation, wikification and information retrieval. Such applications will also enable deeper analyses and better understanding of the strengths and limitations of LOTUS. These analyses can include looking into knowledge sources to help infer implicit information about the literals, such as language tag (when missing) and topic domain.

We expect LOTUS to grow with its applications. Here, we foresee the need for new and more robust string matching functions (for instance, by introducing

fuzzy matching based on Levenshtein edit distance [9]) and we expect challenges in scalability and response time. Additionally, integration with structured data, for instance via SPARQL endpoints, is another direction we aim to investigate.

## References


1. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: Lod laundromat: a uniform way of publishing other peoples dirty data. In: ISWC 2014, pp. 213–228 (2014)
2. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: SPARQL web-querying infrastructure: Ready for action? In: ISWC 2013 (2013)
3. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax (2014)
4. Davis, M., Whistler, K.: Unicode normalization forms (August 2012), `http://www.unicode.org/reports/tr15/tr15-37.html`
5. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 782–792. Association for Computational Linguistics (2011)
6. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An Empirical Survey of Linked Data Conformance. Web Semantics: Science, Services and Agents on the World Wide Web 14, 14–44 (2012)
7. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep., DTIC Document (1996)
8. Kittur, A., Chi, E.H., Suh, B.: What's in wikipedia? : mapping topics and conflict using socially annotated category structure. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09). pp. Pages 1509–1512 (2009)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
10. Ling, X., Singh, S., Weld, D.S.: Design challenges for entity linking
11. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. pp. 1–8. 7th International Conference on Semantic Systems. ACM (2011)
12. Moro, A., Raganato, A., Navigli, R.: Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics 2, 231–244 (2014)
13. Navarro, G.: A guided tour to approximate string matching. ACM computing surveys (CSUR) 33(1), 31–88 (2001)
14. Phillips, A., Davis, M.: Tags for identifying languages (September 2009), `http://www.rfc-editor.org/info/rfc5646`
15. Prud'hommeaux, E., Buil-Aranda, C.: SPARQL 1.1 Federated Query (2013), `http://www.w3.org/TR/sparql11-federated-query/`
16. Rizzo, G., Troncy, R.: Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In: Proceedings of EACL 2012. pp. 73–76 (2012)
17. Tjong Kim Sang, E.F., Meulder, F.D.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Proceedings of CoNLL-2003. pp. 142–147. Edmonton, Canada (2003)
18. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Proceedings ISWC'07/ASWC'07. pp. 552–565 (2007)