



# FUSEPOOL

## STREP

## FP7 – 296192

# D4.2 SEMANTIC SEARCH API WITH ONTOLOGY LINKING

**COORDINATOR: THOMAS GEHRIG**

**LEAD AUTHOR: GUILLAUME BOUCHARD**

**CO-AUTHOR(S):**

**JULIEN PEREZ**

**ADRIEN CZERNY**

**JOHNATHAN REY**

**1ST QUALITY REVIEWER: ANTON HEIJS (TREPAREL)**

**2ND QUALITY REVIEWER: SARVEN CAPADISLI (BUAS)**

| | |
|---|---|
| Deliverable nature: | Report (R) |
| Dissemination level: (Confidentiality) | Public (PU) |
| Contractual delivery date: | 01/04/2013 |
| Actual delivery date: | 10/08/2014 |
| Version: | 6.2 |
| Total number of pages: | 34 |
| Keywords: | Fusepool, ontology management, link prediction, machine learning |

# Abstract

This document describes the Fusepool semantic search API to enable semantic queries, i.e. queries on entities that are not necessarily terms in documents, but can be person, organizations, concepts, etc. Also, even for documents, semantic queries enable a user to improve the search by using non-textual content, such ias document meta-data. The OSGi module responsible for handling the semantic search API is presented, with the details of the underlying link prediction method that is used. The developed software architecture of prediction is very similar to the one developed for the document labelling task (Deliverable 3.2) and makes use of a component called “Listen-Update-Predict” (LUP) that enables a quick and safe integration of adaptive components in the platform.

We present a factorization model with the underlying assumption that any value of the knowledge base (observed or not) is a dot product between latent vectors. These vectors (or embeddings) are computed offline by an incremental algorithm that minimizes a loss function between the model prediction and the observed values of the knowledge base. Numerical experiments illustrate the principle of this approach and a new probabilistic framework for queries is described.

# Executive summary

This deliverables summarizes the work that has been done in the task T4.5 to enable semantic search and handle user feedback. The goal of T4.5 is to enable the Fusepool platform to automatically search and annotate entities in a knowledge base (i.e. a relational database) with missing, incomplete or noisy information. This functionality is a strict generalization of the work done in the document labelling task (T3.4) where the user could label a documents and search for documents having a given label. The extension enables the labelling of any entity such as person or organization, not only document. In addition, the search does not only use the textual content, but also takes as input any meta-data available in the knowledge base, such as the date, the author, the owner of the patent, the publication organization of a scientific document, etc.

In addition to the machine learning framework specified in Task 4.1, implemented in Tasks 2.5 and 3.4, we extended the *PredictionHub* OSGi bundle that centralizes the access to any *Listen-Update-Predict* (LUP) module available on the platform. These three functionalities have been implemented for this sematic search adaptive prediction task 4.5.

Finally, for the updating functionality of sematic search, a factorization model of the knowledge base is proposed. It enables us to infer the most relevant entities associated to each search result when displayed on the search result list of a user.

# Document information

| | | | |
|---|---|---|---|
| FP7 Project Number | 296192 | Acronym | Fusepool |
| Full Title | Fusepool – Information pooling for product/service development and research | | |
| Project URL | http://www.fusepool.eu | | |
| Document URL | | | |
| EU Project Officer | Martina Eydner | | |

| | | | | |
|---|---|---|---|---|
| Deliverable | No. | D4.2 | Title | Semantic search API with ontology linking |
| Work Package | No. | WP4 | Title | Knowledge finding and matching |

| | | | | |
|---|---|---|---|---|
| Date of Delivery | Contractual | M20 | Actual | M18 |
| Status | Version 1.0 | | Final □ | |
| Nature | Prototype □ Report X Dissemination □ | | | |
| Dissemination level | Public Consortium x | | | |

| | | | | |
|---|---|---|---|---|
| Authors (Partner) | Xerox, Searchbox, Geox, Buas, Treparel | | | |
| Lead Author (WP Lead) | Guillaume | Bouchard | Guillaume.Bouchard@xerox.com | |
| | Xerox | | | |

| Version Log | | | |
|---|---|---|---|
| Issue Date | Rev. No. | Author | Change |
| 20 April 2014 | 1.0 | Guillaume Bouchard | First version |
| 5 May 2014 | 1.1 | Adrien Czerny | Description of the semantic search LUP |
| 7 May 2014 | 2.0 | Guillaume Bouchard | Updated version with intro, summary and conclusion |
| 15 May 2014 | 3.0 | Julien Perez | First description and experiments of p-sparql |
| 16 May 2014 | 3.1 | Guillaume Bouchard | Detailed description of p-sparql |
| 17 May 2014 | 4.0 | Jonathan Rey | T4.4 contribution |
| 18 May 2014 | 4.1 | Guillaume Bouchard | Clean-up and reformatting |
| 17 June 2014 | 4.2 | Guillaume Bouchard | Add screenshots |
| 30 July 2014 | 5.0 | Guillaume Bouchard | Took reviewer feedback into account |
| 31 July 2014 | 6.0 | Guillaume Bouchard | Final Check |
| 8 Aug 2014 | 6.1 | Anton Heijs<br>Guillaume Bouchard | KMX paragraph added |
| 9 Aug 2014 | 6.2 | Thomas Gehrig | Final Check |

# Table of Contents

# Abbreviations

| Acronym | Full meaning |
|---|---|
| Annostore | Repository containing annotations |
| API | Application programming interface |
| BUAS | Bern University of Applied Sciences |
| ECS | Enhanced Content Store |
| DoW | Description of Work (part of the Grant Agreement) |
| ENOLL | European Network of Living Labs |
| GEOX | Geox KFT |
| GUI | Graphical User Interface |
| Hitlist | List of documents output from the search result |
| LL | Living Lab |
| LOD | Linked Open Data |
| LUP | Listen-Update-Predict |
| MS | Milestone |
| OSGi | Open Services Gateway initiative |
| RAD | Rapid application development |
| RDF | Resource Description Framework |
| SEARCH | Searchbox SA |
| SME | Small and Medium Enterprise |
| TREPA | Treparel Information Solutions BV |

| | |
|---|---|
| WP | Work package |
| XEROX | Xerox SAS |
| XSLT | eXtensible Stylesheet Language Transformations |
| XML | Extensible Markup Language |

# I. Introduction

Classical information retrieval systems generally focus on search of documents with textual content, by matching keywords to a query. However, with the development of large scale non-textual knowledge bases following the Linked Open Data (LOD) principle, there is today a need for searching other types of resources, such as people, organization, concepts such as events, diseases which might not be easily represented or identified by a simple textual description, by associated attributes that match most probably with the entity of the semantic search" or does it make the sentence to long and complex. For example, people can be represented by their social connections, their past professional achievements and metadata such as age or eye colour. We therefore need for search functionalities that go beyond simple textual search, and enable us to query entity types that are well identified. In Fusepool we concentrate on the seach for people and organization, with a dedicated visualization that is automatically adapted to the information available for a given entity type, using the Uduvudu framework developed in WP5. In the following we describe architecture to enable an efficient semantic search.

The goal of T4.4 and T4.5 is to enable the Fusepool platform to automatically retrieve entities when presented in search result list. This task will naturally be used to enhance the user experience associated to the retrieval results. Figure 1 presents the workflow and dependencies implied by the machine-learning framework introduced in D.4.1 as a general approach to account for user feedback in the Fusepool platform.

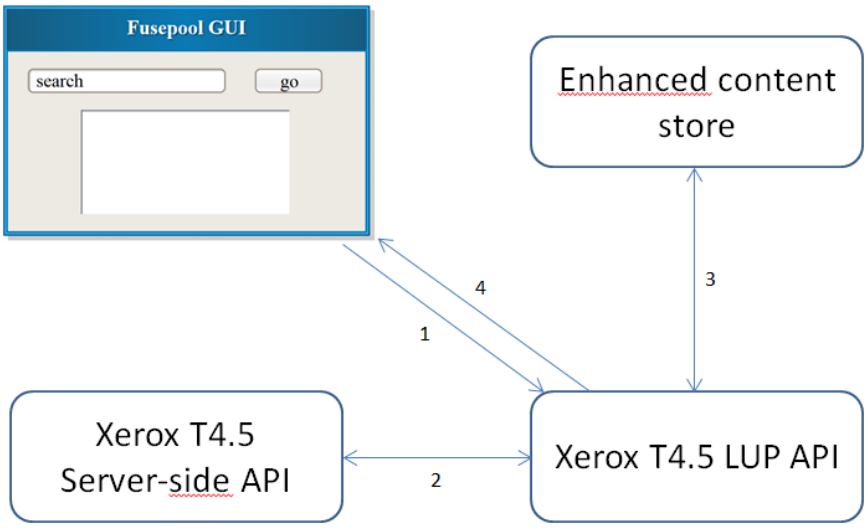


**Figure 1. Overview of the Fusepool machine learning framework components for semantic search.**

In short, the workflow for our semantic search API applied to head-hunting can be detailed by the following steps:

1. The user launch a search in the interface, such as the Firstswim application, which can be defined as the Fusepool GUI, and then the query words are transmitted to the Xerox T4.5 LUP API.
2. The Xerox T4.5 LUP API transmits the query in order to obtain the most relevant Author entity identifier with respect to the query.
3. Using the graph in the enhanced content store, the Xerox T4.5 LUP module forges the result knowledge graph and the corresponding facets that are meta-data of the result graph. The graph contains the Authors information and the related corporation and documents.
4. The Xerox T4.5 LUP retrieves the result graph to the GUI component of Fusepool for display. In the current implementation, the learning framework model used by the author prediction engines is a multi-linear statistical relational model. Moreover, the software design and implementation aspects of this work will be presented and justified.

This document is organized as follows: first, we present an analysis of current semantic indexing solutions with Solr, the main Fusepool software component providing search engine capabilities. Then, a detailed description of a novel probabilistic framework for predicting missing semantic information is made. Finally, we give description how this framework has been implemented in Fusepool.

# II. Solr technology for semantic search

This section describes the work done in Task 4.4 focusing on analysis of the most effective semantic indexing technologies available with an existing indexer such as Solr. It demonstrates several key features of the improvement made by Searchbox to demonstrate the feasibility or large scale semantic indexers.

## Introduction

Information workers are struggling more than ever to sort through mountains of information. As technology becomes easier and cheaper for creating and storing information, the world has been experiencing a data boom, commonly referred to as big-data. While the benefits of additional knowledge are obvious, finding that knowledge is becoming more of searching for a needle in a hay stack. This searching, or researching, is taking more and more time each year while at the same time is becoming less and less assuring that all pertinent information is found.

This section compares two real-time approaches to the related search paradigm: MoreLikeThis, a keyword based approach provided with Solr, versus SenseLikeThis, a conceptual search approach developed by Searchbox. In both cases, the algorithms provide a set of documents in response to a query document consisting in a set of searched terms. This is similar to approaching a librarian, giving them an abstract detailing a concept and asking them to find additional information.

We argue that semantic search is often superior for information recall as opposed to keyword searching. We further discuss the known bottlenecks, which Fusepool technology has overcome. In conclusion we discuss a survey of experts which models a real-world use case. Briefly, we find that survey participants were more satisfied with this technology, which is more able to identify a robust signal in the query document, versus Solr's standard approach, motivating the usage of this new technology.

## Challenge

In a study from 2005, (Bergman, 2005) it is noted that in the advanced knowledge economy of the United States, the information contained within documents represents about a third of total gross domestic product, or an amount of about $3.3 trillion annually. Estimates indicate that corporate data doubles every six to eight months, 85% of which are documents (Grimes). Additional key-points are highlighted by the infamous Coopers & Lybrand report from the late 1990's:

- 90% of corporate memory exists on paper
- Out of pages that get handled in the office, 90% are merely shuffled
- Companies spend an estimated $20 in labour to file a document, $120 in labour to find a misfiled document, and $220 in labour to reproduce a lost document
- 7.5% of all documents get lost, 3% of the remainder get misfiled, a total 10.5% of problematic documents
- Professionals spend 5-15% of their time reading information, but spend 50% looking for it
- A typical worker spends thirty minutes to two hours a day searching for documents

These problems only become worse as the amount of data and number of data sources increase.

The question that we discuss here is: how can we find information that we have hidden in our document repositories faster, cheaper and with greater accuracy and completeness (i.e. precision and recall)? In this paper we consider two different paradigms, the familiar keyword search paradigm and an exciting new implementation of conceptual search, which Searchbox has developed specifically for enterprise document discovery.

The paradigm made popular by Google is the retrieval of documents by keyword. If the user has read a document and wishes to find similar documents, it is their responsibility to reduce the document to keywords and iteratively search through the combinations, and their synonyms, until they find the answer they require. This is an excellent approach when the user is only looking for an answer via a single document, and knows with high certainty when they've read the answer. Examples are: "Who is the president?", "What day of the week is Christmas 2012?", "Example of add document in Solr".

The idea of *conceptual searching* is in fact similar to the way people naturally search for information. Conceptual searching is the heart of the definition of research: "The systematic investigation into and study of sources in order to establish facts and reach new conclusions." When we want to look at a *set* of documents, which are related in *overall idea*, not simply on a keyword level, and gather their combined knowledge, conceptual search is the answer.

For example, when searching via keywords, if the query is "teenager", it will not produce results containing the word "adolescent", which is equally as valuable. Keyword search is excellent when searching for "an answer", but the conceptual searching paradigm is the next step for information workers searching for "the answers". We like to think of it as having your own personal librarian.

This idea is explained very clearly by the Latent Semantic Indexing Wikipedia page: *"LSI overcomes two of the most problematic constraints of Boolean keyword queries: multiple words that have similar meanings (synonymy) and words that have more than one meaning (polysemy). Synonymy is often the cause of mismatches in the vocabulary used by the authors of documents and the users of information retrieval systems. As a result, Boolean or keyword queries often return irrelevant results and miss information that is relevant."*

Typical use cases, which easily differentiate the two paradigms, are:

Legal Studies: The user has a case summary for which they'd like to find all related legal precedents. The number of ways to keyword search for *all* related information is daunting; with a conceptual search a simple copy and paste does all the work.

Patent Studies: Trying to find patents, which your idea infringes on, can be a challenging task. Conceptual search allows a user to copy and paste their abstract and find related patents within seconds.

Research References: Looking to publish but are unsure if the important related works are cited? Provide the abstract and conceptually related documents are listed for quick review.

## Proposed solutions

There are automated approaches for each paradigm. In particular, we would like to provide a document and discover all of the documents, which exist inside of the repository which are similar and might provide additional information into the idea we're researching. Another interesting application is of course the discovery of identical documents, or slightly modified documents, easing the identification of various versions across a content management system.

We discuss two approaches, which are implemented in Solr. Taken from http://lucene.apache.org/solr/:

Solr is a popular search engine solution. Today, this is the largest open source project in this domain.It is based on the Apache Lucene indexer. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is reliable, scalable and fault tolerant. It can provide distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. One of the major advantage of Solr is that it has an extensive plugin architecture when more advanced customization is required.

**MoreLikeThis (MLT) Search**

MLT is included with default Solr installations. Its request handler takes a Lucene query and produces a set of documents, which it determines are most relevant to the query document. This approach is

open source and there is a good explanation of it at http://cephas.net/blog/2008/03/30/how-morelikethis-works-in-lucene/, therefore we onloy provide a brief summary of its algorithm and some pros and cons to its usage.

MoreLikeThis essentially functions like an automatic keyword extractor. Given a query document, the algorithm proceeds by identifying "interesting terms" and using a set of Boolean queries to identify documents in the repository, which contain some favourable combination of them. This is to say that if the terms, which are identified as "interesting" are incorrect the result set will be unfortunately unrelated.

The way in which the terms are selected is based on an approach known as term frequency-inverse document frequency, or tf-idf for short1, used to provide a weighing for how important a word to each document is. The selection proceeds by creating a score for each term by considering the two parts. Term frequency is a numerical value for the number of times a term appears in a document, this gives some general idea of what is important (for example: try writing a paragraph about dogs without mentioning the word dog a few times). On the other hand, the inverse document frequency component identifies terms which are unique as compared to the corpus, and thus have the potential for differentiation (for example: if all 10,000 documents in your repository contain the word dog, dog no longer becomes "interesting".). The way to obtain a high score is thus to appear in a single document very often, but appear in the rest of the documents infrequently. A more thorough mathematical explanation is available here: http://en.wikipedia.org/wiki/Tf-idf.

Pros

| Item | Description |
|---|---|
| **Easy installation** | Simply enabling the request handler gives good performance |
| **Low overhead per query** | Low hard drive and memory usage per query resulting in high throughput on even weak machines |
| **Dynamic relative to repository** | As documents are added and removed from the repository the query terms are automatically adjusted to be more "relevant" |

Cons

| Item | Description |
|---|---|
| **Unintelligent selection of keywords which define query document** | Selection of the terms is done without contextual information of underlying word meaning. This makes it easy to mis-understand a document and return un-related results |
| **Lack of synonym search** | Searching with a document that describes “teenagers” will never return a document which discusses “adolescents”, i.e. query terms are limited to terms which are present in document |
| **Same Query -> Different results** | While dynamic updating is beneficial, it often as negative consequences for information workers. This is seen as remember *how* to get the answer (it’s the 5$^{th}$ result when this document is a query), but not the actual answer itself. |

**Search based on distributed representations**

Distributed representations corresponds to the use of vector-based document representations. Instead of representing a document using a set of words, this implies that an algorithm learns this vector-representation using a statistical method. Here, we present the most popular of these distributed method: the Latent Semantic Analysis (LSA). LSA creates a matrix model via a truncated Singular Value Decomposition (SVD) of a document term frequency matrix (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). This model creates a semantic space within which projected documents having similar concepts are closer to each other, allowing for accurate retrieval and ranking of related documents.

Intuitively: A model is created which has learned the meaning and relationship between words by examining all of the documents in the repository. By supplying this domain knowledge, we are able to identify terms/concepts that are similar in their underlying context automatically. This is no different than reading many books on a particular subject, forming the relationships of words and ideas in one’s mind, and then through that lens identifying related documents to a query.

While this approach has been well known and studied in academia for many years, showing superior results to keyword retrieval [ (Husbands, Simon, & Ding, 2000), (Dumais, 2003)], there were numerous hurdles preventing it from entering a production environment:

Computation of latent representations: The creation of a good model for small datasets has been well studied [ (Berry, Dumais, & O'Brien, 1999), (Kumar & Srinivas, 2006), (Hofmann, 1999)], unfortunately, as the datasets have grown in size (consider PubMed with 11M documents), the computational cost rises exponentially as the number of documents and additional terms grows with the potential for reaching into weeks with even a powerful 12 core machine.

Memory Requirements: As greater specificity is required for larger homogenous datasets to identify the optimally related documents, the size of the model must also increase putting stress on even today’s high memory instances. At this level, trying to hold large datasets in memory is completely intractable.

Efficiency of search: The larger the model, the longer the search time will take as each document needs to be projected. Additionally, since production size datasets are too large to hold into memory, retrieving documents from disk also creates significant latency in requests. For moderately large datasets of 100k, search times reached into the 1 minute range, completely unusable for any production environment.

Software Framework: The development of an entire enterprise level software solution (including the CRUD of documents, data import handlers, and necessary features for enterprise usability such as

highlighting, spell checking, etc.) was an insurmountable task for small startups, let alone integrating conceptual search algorithms.

Over the last 3 years, the Searchbox research team has developed proprietary cutting edge algorithms that solve the aforementioned problems. Furthermore, the technical team has developed a Solr plugin, Searchbox-Sense, which brings the previously unavailable power of semantic search immediately into an existing Solr installation with good results.

As one might have determined, the power of the conceptual search comes from the latent representation. With this plug-in we provide a specific latent representation for the included demonstration dataset, thus providing very high relevancy in its domain. This latent representation can of course be used for evaluation on different datasets, but typically we see about a 15% degradation of result quality as a consequence of out-of-domain knowledge errors (e.g. don't ask a doctor how to fix a car).

Pros

| Item | Description |
|---|---|
| **Easy installation** | Simply enabling the request handler gives good performance |
| **Deeper conceptual search** | Documents which match the idea being spoken about, as opposed to a small finite set of keywords, are found and returned, reducing search time and improving information discovery. |
| **Automatic synonym expansion** | By automatically accepting synonyms, results are more robust as document signal is enhanced. |

Cons

| Item | Description |
|---|---|
| **Additional overhead** | While not overwhelming, each query is a bit more complex than a normal MLT query requiring middle level hardware. |
| **Only a CKB specific to the dataset generates fantastic results** | While a generic latent representation can do well, the true meaning of the terms used in your documents are best learned from the documents themselves, implying a training process. Thankfully these models, once created, have a very long shelf life. |

## Case Study

### *Technology Setup*

On a small Amazon EC2 cloud instance, we installed Solr 4.0 and enabled the MLT request handler. Following the instructions for the installation of SLT (see Fusepool Github repository), we enabled the SLT request handler. In both cases installation was fast and took under 5 minutes.

### *Data Setup*

Searchbox has been granted a license to the dataset of Medline, the full dataset of Pubmed. This dataset consists of over 19 million articles from over 4,800 indexed titles (journals, conferences, letters, etc) encapsulated in 83 GB.

The information is presented in 685 xml files, which contain multiple articles. An XLT conversation was used inside of the Solr data import handler to extract each article, and index the relevant fields which include author, title, abstract, keyword, journal name, date, etc.

### *Experiment Setup*

We asked experts in various research fields to identify Medline articles which they had read in the past. These fields included psychology, neuroscience, biomedical imaging, business, and development frameworks. We provided 2 sets of 10 results, 1 created by MLT and 1 created by SLT and asked them to identify which set of results were more relevant to their research. Comparisons use the abstracts as most articles were closed-source. We constrained each query for both algorithms such that it would take less than 2 seconds, i.e. within usability for real-world enterprise use.

### *Results*

Highlights

- Participants were overall more satisfied with the results produced by SLT than MLT, re-enforcing the published works mentioned in earlier sections and motivating the adoption of conceptual search technology.
- Some experts were exposed to documents via SLT which they were unaware of, but are now eager to read knowing of their existence.

**Specific Examples**

Of course a quick review of some specific examples will further drive home this point in a concrete fashion. Find below some sample results from various queries from different fields, comparing MLT to SLT.

**Business**

Query: Beyond corporate strategy.

MLT Result: A Baby Bell reexamines itself. Decision support systems help planners hit their targets.

SLT Result: Taking strategy to the bottom line. The generic strategy trap.

We can see that MLT is able to identify that corporate is a valuable term, unfortunately The Baby Bell abstract demonstrates how a simple tf-idf extraction can lead to false positives. Since the term corporation is one of about 20 terms, it has a high weighting, encouraging its documents retrieval, when in actuality the document is very sparse.

On the other hand, we can see that SLT understands that corporate is important with its relation to the term strategy and thus is able to pull out the most relevant documents to the query.

**Information Technology**

Query: Alignment of biological sequences with quality scores.

MLT Results: Integer programming-based approach to allocation of reporter genes for cell array analysis.

UPNT: Uniform Projection and Neighbourhood Thresholding method for motif discovery.

SLT Results: Optimised fine and coarse parallelism for sequence homology search.

Unique marker finder algorithm generates molecular diagnostic markers.

We can see that “sequence homology” and “alignment” are represented in a similar fashion for SLT, producing pertinent results. On the other hand, MLT struggles to differentiate cell arrays from the underlying concept of DNA, most likely because it was identified incorrectly as a term of interest.

**Biomedical Imaging**

Query: A magnetic resonance spectroscopy driven initialization scheme for active shape model based prostate segmentation.

MLT Results: Interactive segmentation of abdominal aortic aneurysms in CTA images.
SLT Results: Segmenting the prostate and rectum in CT imagery using anatomical constraints.

We can see that while both algorithms produce results related to segmentation, only SLT is capable of also identifying results that are related to the prostate…making them more similar to the query document. This is due to the non-discrete nature of the SLT, and not forcing the identification of only a few terms, but instead allowing each term to contribute to the overall concept.

**Neuroscience**

Query: Serotonin modulation of cerebral glucose metabolism in depressed older adults.

MLT Results: PET in generalized anxiety disorder.

SLT Results: Depression in schizophrenia: II. MRI and PET findings.

Quite simply the SLT identifies the concepts of depression for its result set, while MLT is likely identifying the PET concept presented in the query document. As a result, the wrong concept is identified and the results that are presented are less relevant. Again, because SLT allows each term to complete a piece of the overall picture, results are more specific to the overall theme as opposed to individual facets that may or may not be correct.

**Additional Comparisons**

We find that often the first few documents in each result set are the same, but that they start to differ later on. As a quick reference we compare 8 queries using MLT and SLT and show the average percentage of documents that are the same below:

| | |
|---|---|
| Top document in common | 50.00% |
| Top 5 documents in common | 35.00% |
| Top 10 documents in common | 31.25% |
| Top 25 documents in common | 36.50% |
| Top 50 documents in common | 36.50% |
| Top 100 documents in common | 34.75% |

In our Searchbox-Sense package, we provide a python script that performs these comparisons automatically. This script is available in Fusepool github repository.

# Conclusion

The semantic search API enables each user to predict links between entities in a knowledge base in a way that search result lists become enhanced with richer kind of information concerning the retrieved entities. Preliminary tests already provide encouraging results in an experimental setup. There are several ways it could be improved and in particular the way entities are represented and the type of factorization used.

Similarly to D3.2, this work also illustrates the generality of the Fusepool learning framework defined in D4.1, which enables us to quickly defined new learning components based on user feedback, without having to redefine the whole data lifecycle pipeline including user feedback, update of the model parameters, version updates and backup functionalities.

We also have demonstrated how semantic technology allows for a more relevant, robust approach towards finding relevant documents given a query document over another standard industry approach. By identifying a more robust signal for a document, learned from an enterprise’s own repository, the system is able to deliver real-time results to information workers who are wasting hours of time in attempts to research topics.

# III. Semantic Prediction in Fusepool

This section presents some research done in the context of Fusepool, after remarking that many semantic web data were noisy or had many missing elements, and would therefore need a generic way to predict missing elements. We first introduce and explain the concept of predictive queries in RDF, called P-SPARQL. Then the core learning algorithm based on low-rank tensor approximation of a knowledge base is presented with experiments.

## Probabilistic SPARQL

During the last decade, significant progresses have been accomplished in the field of automatic extraction and representation of knowledge from structured and unstructured data sources like databases or texts. Many institutional data sources, available on the Web, are today organized using the RDF format and accessible via SPARQL. In parallel, the research field of statistical relational learning has known, mainly these last five years, a rapid growth of interest linked to the progress obtained in the field of the factorization of tensors and in particular its application in the approximation of the missing values.

During Fusepool first year, we thought about an original SPARQL request engine based on a probabilistic modelling of the underlying knowledge base. It happens to be a generic way to query information, and we called the resulting system P-SPARQL, for Probabilistic SPARQL. In the following, we formalize, design and implement a SPARQL 1.0 engine that is based on querying information from a low-rank tensor approximation of a source RDF knowledge base. The system will be seamlessly able to incorporate confidence calculus in each result triple. This architecture will also allow the user to predict missing relation between pairs of entities while using the standard SPARQL syntax. As a last side result, the system will also allow the user to detect outliers and suspicious results in the knowledge base.

Knowledge bases stored using the RDF format of triples are popular data structures used today for formal modeling of the relations between entities. Recently, the question of querying languages for this sources of data has attracted the attention of several scientific communities, including database management, but also statistical learning and formal logic. In this context, the W3C developed SPARQL as query language for RDF datasets. RDF storage systems supporting this query language has been developed such as Jena, sesame, rdfdb, redland, kowari, RDFsuite and Allegro. The wide adoption of this language makes a de facto standard for querying Linked Open Data. In addition, in January 2008, SPARQL has officially been approved by a recommendation of the W3C, a status that no other RDF query language reached until now. SPARQL supports the requesting of triple patterns, conjunction and disjunction. The results of a request can be ordered and limited. RDF Data are organized in triples of the form (S,P,O), where S is the subject, P the predicate and O the object of the triple. The result of a request is a set of tuples aggregated after extraction of the sets of triples SPARQL matching the sequence of clauses of a SPARQL request. In fact, this formalism takes its inspiration within the formal framework of description logic. At present, the SPARQL request engines are mainly based on two paradigms: ensemblist algebra and pattern search in graphs.

To our knowledge, there is no solution today to obtain a well-defined measure of data uncertainty together with SPARQL results, despite the large amount of probabilistic extensions to RDF, such as the effort to bring machine learning algorithms within the SPARQL language using statistical relational learning. Indeed, although some works tackles this question, no SPARQL request engine per se proposes a formal calculation of the probability of a triple to be valid or a list of results ranked by probability. Furthermore, the induction of missing data, as far as we know, has never been proposed in a SPARQL engine and is left for reasoning engines such a Pellet. Beyond the representation of the knowledge base as a graph, one can regard each predicate as a partially observed adjacency matrix between subjects and object entities. Thus, it appears interesting to use information already available in the KB in order to induce, in statistical sense, the probability of correctness of unseen triples.

The Predictive SPARQL engine work as follows:

1. a user write a formal SPARQL query that he would like to run on his KB. Unlike standard queries, the user can assume that the KB is perfect, i.e. it contains all the possible information. For example, one can query “what is the weather tomorrow” or “what is the category of this document”, even if the targeted document has not been annotated yet.
2. The P-SPARQL engine parses the query exactly the same way a standard SPARQL engine would do, but the clause processing is completely different: instead of keeping a set of results for each clause, a probabilistic distribution on the elements within the clauses is computed.
3. Then, during the query interpretation, probabilistic calculus is used to obtain the final results. For this calculus to be valid, we use two main assumptions: (i) each clause is independent to each other conditionally to the KB distribution and (ii) the prior probability of entities is known (usually assumed to be uniform).

The overall process is illustrated in .

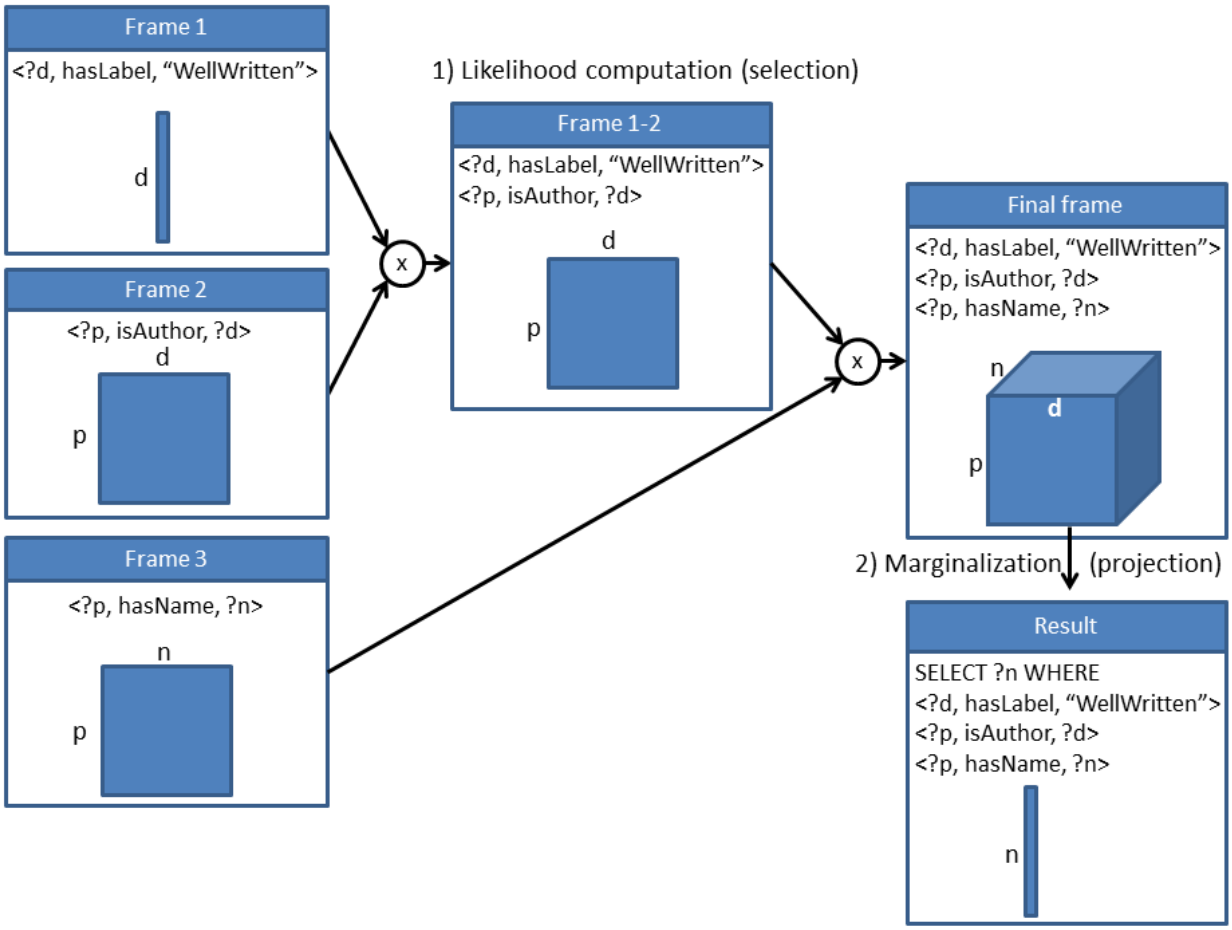


**Figure 2. Illustration of the probabilistic clause processing in P-SPARQL. The query on the right is decomposed into elementary frames on the left that are directly computed based on the probabilistic representation of the database. The joint operations correspond to probabilistic rules (multiplication in this case to represent the AND operator). In the final step, the selection corresponds to the addition of the probabilities that are presented by a tensor (a third order tensor in this case because the query contains 3 unknown variables).**

## Overview of the Predictive SPARQL architecture

To get a high level understanding of the predictive queries described in the previous section, we give in Figure 2 a flowchart showing on the left a standard SPARQL engine that uses a triplet store (Observations D) to run queries, and usually retrieves a set of results. However, in many applications, the data are missing and (for example the category of a new document, or the location of a tweet that is not geo-localized). The predictive query principle that we proposed consider a probability distribution over all the possible entries, missing or observed. In the flowing section, we describe one possible model for this probability distribution. The results are supposed to give similar predictions as the standard query if there is no noise in the data, but might predict different values if the data are noisy (e.g. user ratings), so that predictive SPARQL can be used as a tool to detect outliers by comparing predictions to actual observations. For missing data, the predictive SPARQL is a powerful tool that can give confidence intervals on the result set, enabling a natural ordering of them.

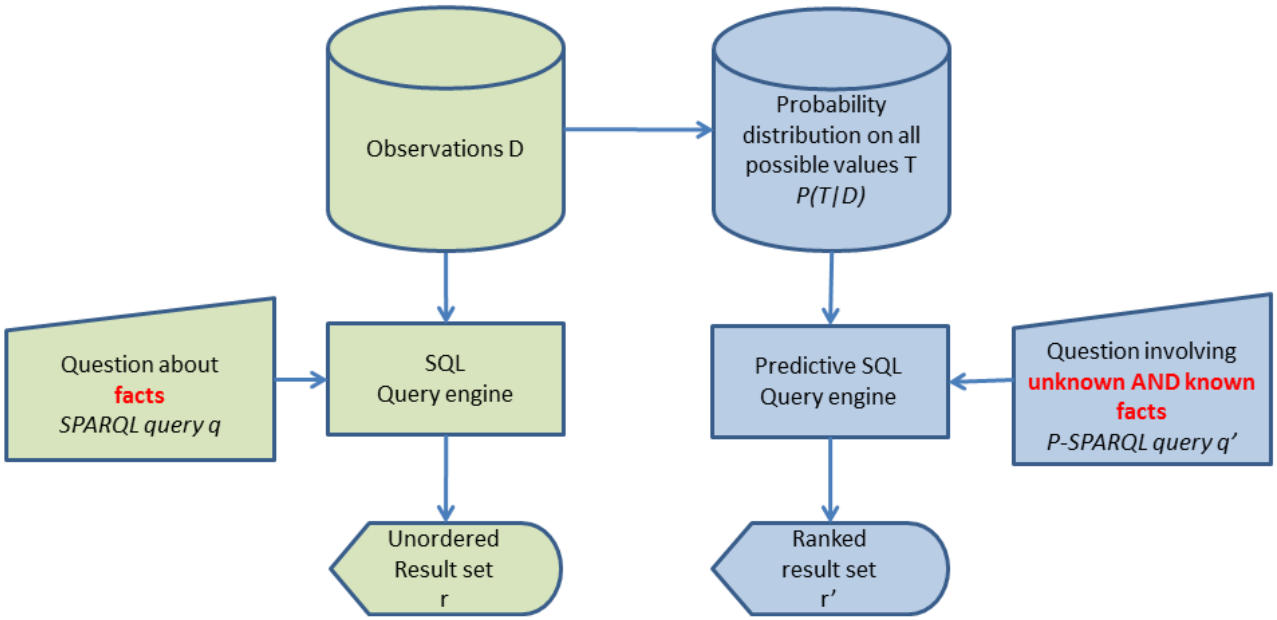


**Figure 2. Overview of the predictive SPARQL engine.**

## A probabilistic model for relational data: knowledge base factorization

In this section, we describe one possible probabilistic model that is compatible with the Predictive SPARQL framework that was just described.

The learning engine is in charge of predicting missing links in a knowledge base. Indeed, exploiting the information contained in the relationships between entities of a knowledge base has been a topic of interested in the machine learning scientific community. For example, social network analysis, bioinformatics make extensive use of relational information, as do large knowledge bases such as Google's Knowledge Graph or the Semantic Web. It is well-known that, in these and similar domains, statistical relational learning (SRL) can improve learning results significantly over non-relational methods. In this context, inference is often based on methods such as MCMC which introduce additional scalability issues. Recently, tensor factorization has been explored as an approach that overcomes some of these problems and that leads to highly scalable solutions. Tensor factorizations realize multi-linear latent factor models and contain commonly used matrix factorizations as the special case of bilinear models. We will discuss tensor factorization for relational learning based on the factorization of a third-order tensor and which has shown excellent learning results.

Tensor factorizations have become increasingly popular for learning structured data such as large-scale knowledge bases, recommendation data or time-varying networks. The success of tensor methods in these is mainly explained by their ability to model, analyse and predict data with multiple modalities. Moreover, due to their multi-linear nature, tensor models overcome limitations of linear models, such as their limited expressiveness, but remain more scalable and easier to handle than general non-linear approaches.

In this section, we will demonstrate the usage of tensor factorization for relational learning in the context of semantic information retrieval of Fusepool. In the following of this section, we will present the software architecture of the Predictive SPARQL engine we develop for predictive querying based on link prediction. Then, the mathematical model used for link prediction used in this task will be detailed. Finally, a set of experiment will be presented in order to assess the functionality of the module in the previously presented context of headhunting.

**Architecture**

The learning engine associated to the Task 4.5 consists in estimating the joint probability of a knowledge base with missing links. Figure 9 describes the overall interaction that take place in Task 4.5. The T4.5 Xerox Server Side infrastructure communicates with the Fusepool OSGi platform by a HTTP/REST interface that is in charge of three atomic tasks (1) transmitting triples to the server side (2) querying relevant persons based on topic sent as string (3) predicting most probable person's entity for a given query. Having the entity transmitted to the server side will allow having a potential centralization of different corpora associated to several instances of Fusepool. In this context, it would be possible to fusion knowledge generated from user communities involved in different instances of the Fusepool system.

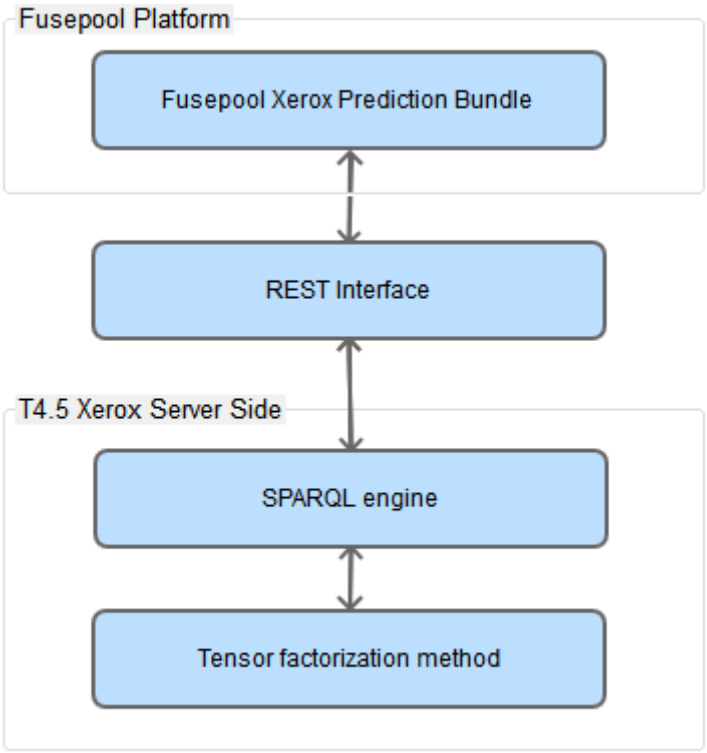


**Figure 3. Interaction between components in the prediction engine**

The actual specification of this Server Side REST interface is the same than the LUP4.5 presented in the previous section. The T4.5 Server Side is decomposed in two modules: (1) the triples of the knowledge base are stored in a matrix format (2) the tensor factorization algorithm that will be presented below (3) the query engine is an implementation of SPARQL 1.0 which is the state of the art language for knowledge base querying.

**Formalization**

Rescal is a tensor factorization model for dyadic multi-relational data which has been shown to achieve state-of-the-art results for various relational learning tasks such as link prediction, entity resolution or link-based clustering (Nickel, Tresp, & Kriegel, 2011). The model can be summarized as following: Let consider a set of relational data with K different dyadic relations and N entities, a third-order adjacency tensor X of size $N \times N \times K$ is created, where :

$$x_{ijk} = \begin{cases} 1, & \text{if } Rel_k(Entity_i, Entity_j) \text{ is true} \\ 0, & \text{otherwise.} \end{cases}$$

This adjacency tensor X is then factorized into latent representations of entities and relations, as:

$$X_k \approx AR_kA^T$$

where $X_k$ is the k-th frontal slice of X. After computing the factorization, the matrix $A \in R^{N \times r}$ contains the so-called latent representations of the entities in the data, i.e. the row $a_i$ holds the latent representation of the i-th entity. Furthermore, $R_k \in R^{r \times r}$ is the latent representation of the k-th predicate, whose entries encode how the latent components interact for a specific relation. Since $R_k$ is a full, asymmetric matrix, the factorization can also handle directed relations. Finally, the symbol $\approx$ denotes the approximation under a given loss function. When learning the latent representation of an entity, unique global representation allows the model to access information that is more distant in the relational graph via information propagation through the latent variables. For example, it has been shown that Rescal can propagate information over multiple relations, such that the correct latent representations are learned (Nickel et al., 2011). Moreover, since the entries of X are mutually independent given the latent factors $A$ and $R_k$, prediction is very fast, as it reduces to simple vector-matrix-vector products.

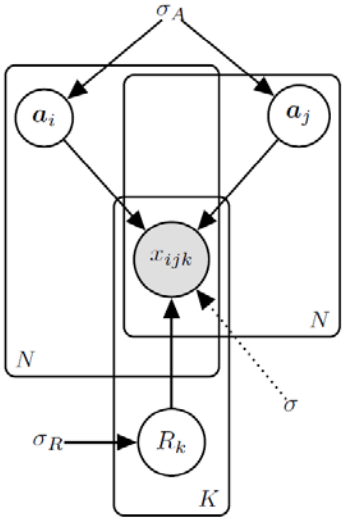


**Figure 4. Graphical model in plate notation of the RESCAL factorization. The parameter σ is only present when the random variable x_ijk follows the normal distribution.**

The model can be interpreted from a probabilistic point of view. Each entry $x_{ijk}$ in $X$ is considered as a random variable and we seek to compute the MAP estimates of A and R for the joint distribution

$$p(\mathcal{X}|A, \mathcal{R}) = \prod_{ijk} p(x_{ijk}|\boldsymbol{a}_i^T R_k \boldsymbol{a}_j)$$

Figure 10 shows the graphical model, in plate notation, for the factorization. We will also define the prior distributions of the latent factors to the Normal distribution, i.e. we set

$$\boldsymbol{a}_i \sim \mathcal{N}(0, \lambda_A I)$$
$$R_k \sim \mathcal{N}(0, \lambda_R I)$$

To learn the parameters, we will maximize the log-likelihood of Equation 1, such that the general form of the objective function that we seek to optimize is

$$\arg\min_{A,\mathcal{R}} \text{loss}(X; A, \mathcal{R}) + \lambda_A \|A\|_F^2 + \sum_k \lambda_R \|R_k\|_F^2$$

The nature of the loss function depends on the distribution that we assume for $x_{ijk}$. We consider the least-squares loss function. In this case, Equation 2 and Equation 3 maximize the log-likelihood of Equation 1 when

$$x_{ijk} \sim \mathcal{N}(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j, \sigma^2)$$

An algorithm based on alternating least-squares updates of the factor matrices, has been shown to scale up to large knowledge bases via exploiting the sparsity of relational data. It should be noted that although the least-squares error has an interesting property that it enables a very efficient and scalable implementation. The scalability of algorithms has become of major importance as relational data is generated in an unprecedented amount and the size of knowledge bases grows rapidly. Rescal is a highly scalable algorithm to compute the Rescal model under a least-squares loss (Nickel, Tresp, & Kriegel, 2012). It has been shown that it can efficiently exploit the sparsity of relational data as well as the structure of the factorization such that it features linear runtime complexity with regard to the number of entities $n$, the number of relations $m$, and the number of known relationship, while being cubic in the model complexity $r$. This property allowed, for instance, to predict various high-level classes of entities in the Yago 2 ontology, which consists of over three million entities, over 80 relations or attributes, and over 33 million existing relationships, by computing low-rank factorizations of its adjacency tensor on a single desktop computer[1]. In the following we will refer to this implementation as Rescal ALS.

**Experiments**

To evaluate our method in the context of the Task 4.5, we conducted link-prediction experiments between authors and subject query based on the knowledge base of the project. More precisely, the task was to predict the pertinence of an author of a document inserted in the Enhanced Content Store of the Fusepool platform. The overall database of triples counts 84.000 triples. The predicates that are considered are:

1. Author
2. Organization
3. Type of document (Patent or Pubmed)
4. Label of document
5. Title-NGram
6. Content-NGram

The *-NGram predicates are produced using the actual text of each document that is tokenized in NGram. Indeed, in the ECS, each document has a unique triple containing the overall text. So, Title-NGram and Content-NGram types of triples are produced in order to obtain an overlap between the triplified descriptions of each document. Figure 11 shows an example of a sentence in a given text.

[1] Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing YAGO: scalable machine learning for linked data. In: Proc. of the 21st International World Wide Web Conference. Lyon, France (2012)

- TextId : *4142*
- Sentence in the text: “It is about using the Web to connect related data that wasn't previously linked”
- Example of triples :

4142 <hasContentNGram> Web

4142 <hasContentNGram> connect

4142 <hasContentNGram> data

4142 <hasContentNGram> related

4142 <hasContentNGram> linked

…

**Figure 5. Example of triplification of the content of a document in the ECS**

In order to evaluate the method, the following protocol has been performed. The goal is to retrieve the correct author with respect to a query generated from a set of words extracted from a given the document where the authors will have been blinded during the learning phase of the model. On the other hand, false author has been added in order to assess the robustness of the process form the same amount that blinded authors.

Following this procedure, a 10-fold cross-validation has been performed. The results are evaluated using the area under the precision-recall curve (AUC-PR) which is a state of the art evaluation method in statistical relational learning kind of tasks. In order to assess the generalization capability of the data, Figure 12 shows results obtained with different values of embedding size.

| Embedding size | AUC-PR |
|---|---|
| 10 | 0.75∓0.06 |
| 25 | 0.85∓0.04 |
| 50 | 0.97∓0.01 |
| 75 | 0.81∓0.02 |
| 100 | 0.78∓0.01 |

**Figure 6. Example of triplification of the content of a document in the ECS**

**Semi-automatic term extraction to generate controlled term list in document clusters**

As part of the optimization task on classification, Treparel has carried out research on the combination of clustering and classification to determine unique relevant descriptive terms for each cluster. Preliminary experiments are very encouraging since it includes an unsupervised approach (LSP algorithm), where documents contain terms that are relevant for a specific sub domain of the data, as well as a supervised approach (classification) applied to every cluster and thus obtain a weighting of the most important documents for that cluster. We have learned that the LSP algorithm in practice capture the concepts of the sub-clusters very well, which became clear from experiments on many different datasets. Off course this is a qualitative observation but an important one. The work then focussed on determining the small number of locally specific terms for each cluster. For this, the algorithm computes for each cluster a ranked list of descriptive terms with weight factors; these terms are then used in the annotation system of the machine learning components of Treparel called KMX. In this specific case we are trying to determine a small set of terms that are unique to the cluster and are very specific and may not even be in known ontologies. We are looking for 10-100 terms specific for a cluster that have a very low term frequency (so in the tail of a local (cluster) ranked frequency distribution).

We have developed code and done experiments in KMX and we are now able to obtain cluster specific terms that we can include in a thesaurus, taxonomy or even an ontology related to the users specific search. Creating these is normally expensive and they need to be maintained since a new research field will see over time new terms describing new concept. In our approach to create from the data controlled term list one can in principle learn from the data these terms. The validation of the approach is on the other hand challenging and here we used data provided from one of Treparel's clients (Bayer Healthcare). They do this work manually and could give us some test data for which they know the unique and rare terms that are very important to do optimized searches. Of course we cannot test our approach on all domains and setting up a validation is already a time consuming task, so we restricted ourselves to the provided (proprietary) biological data.

An implementation on the Fusepool platform can support semi-automatic label suggestion, search expansion suggestion functionality and classification optimization. When we see this in combination with the work presented in the previous chapters, the approach can optimize the labelling effort of the "crowd" when applied to large sets. This work led to good initial results, but it is still unpublished. The planned time (4PM) was not enough to make it a proper integral part of the platform but it is important to determine that in all cases it works as an incremental improvement. Therefore we intend to make the result and the approach public once we have enough testing done. It should be relatively easy to implement this additional functionality in the Fusepool platform.

# IV. Fusepool Integration of Semantic search functionalities

## Implementation of the Learning Framework

The learning framework from D4.1 has been implemented and fine-tuned to task T4.5. We give a short summary of the most important features of the implementation.

Each learnable component hosted on the Fusepool platform is built in order to match a uniform Learning Framework. However, this task needed a few adjustments described below:

1. As data (documents) are inserted in the ECS, two modules (one for each data source) are responsible for Extracting-Transforming-Loading data. A new schema for these data has been defined, and data are made accessible via the REST protocol **(steps 1. and 2.)**
2. These newly generated data are used by a distant server (OpenXerox) to build learning models **(step 3.)**
3. The prediction is enabled on the platform by a module responsible for accessing the learning models on each Semantic Search query **(steps 4.)**

The Figure 7 illustrates the process.

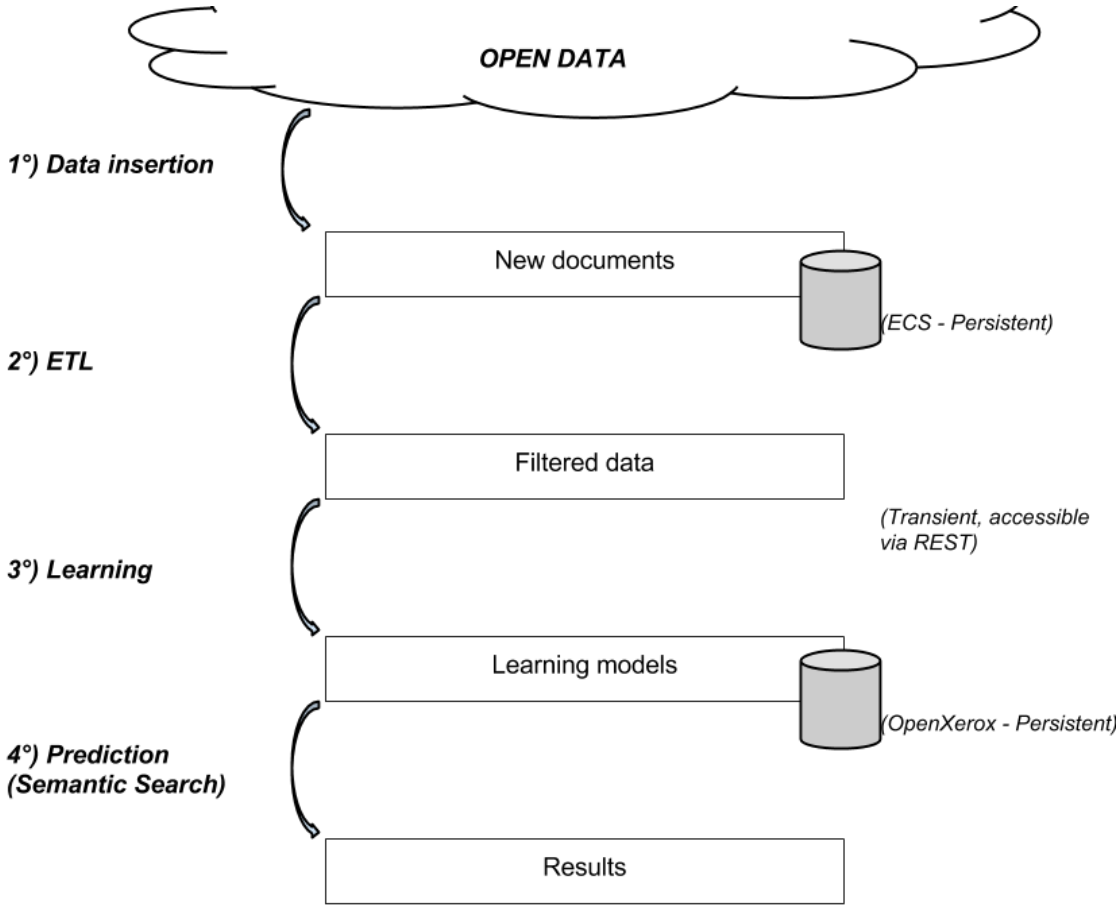


**Figure 7. Learning Framework**

We refer to the D3.2 for a full description of the *PredictionHub* OSGi bundle that is in charge of providing the learning framework infrastructure of the Fusepool platform.

## Extract-Transform-Load for tensor build

A first step for this particular Entity Search feature was to build the models which will be used to xpredict the entity linked to a given query. As you may notice, this feature leads to a very specific use-case of the LUP design pattern since the *Learning* is not triggered by new annotations in the Annostore.

The learning is based on the content of the Enhanced Content Store, after a process of ETL (Extract-Transform-Load) over patents and pubmed documents.

### Refined graph

For both patents and pubmeds document we needed some simple ontologies to represent the title, the abstract, the content, the author(s), the label(s) and the organisation(s) (when available).

On figure 5 you will find a graph representing the data we extract from the ECS.

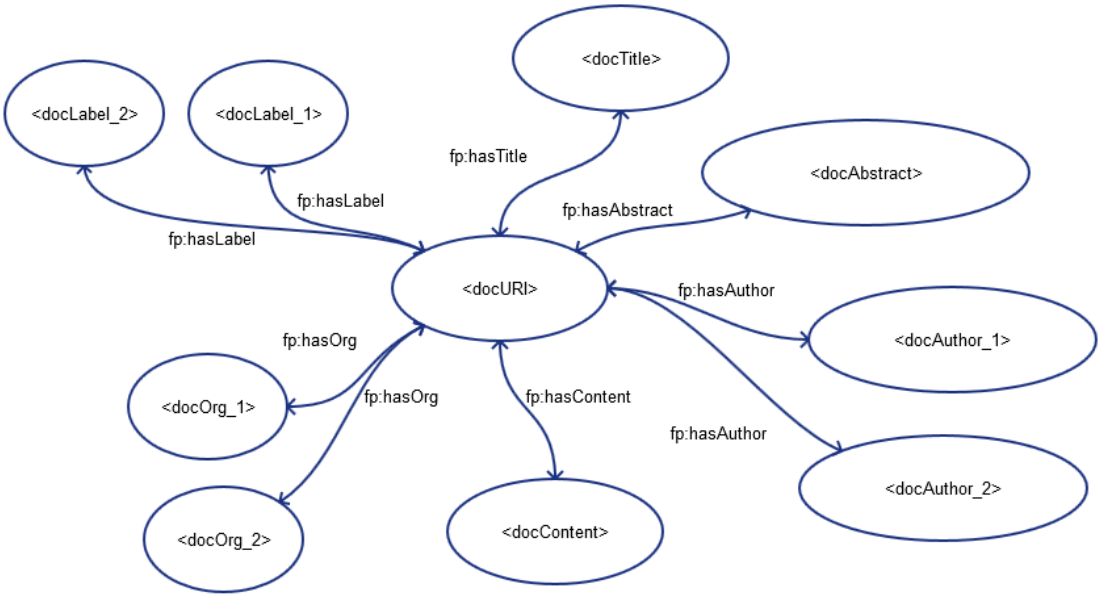


**Figure 8. Simpler data graph used to build the tensor-based models**

These data are not meant to be used by any outsiders but to be used once for building the tensor used for the Semantic Search (c.f. **section 5**); we extended the http://fusepool.info/ ontology to match our needs (using the prefix **fp**) and each predicate refers to a Literal object. For each document, we can find in this graph its:

- Title,
- Abstract,
- Content,
- Author(s),
- Organisation(s),
- Label(s),

**The ETL modules**

Two modules are responsible for extracting and publishing these data:

- com.xerox.patents.etl - **PatentsETL**
- com.xerox.pubmeds.etl - **PubmedsETL**

These bundles proceed to the ETL process at activation and generate the result N-Triple file for these data at the URL:

- /patents/etl
- /pubmed/etl

At activation these modules call the **ETL()** method:

```
void ETL(Iterator<Triple> it, BufferedWriter relationsWriter);
```

… which will parse the *content.graph* to get the data to extract, using Clerezza *graphNodeProvider* component. As the ETL component parses the *content.graph*, it builds a new graph representing the data mentioned above, and serializes it as a Turtle file.

This file will be returned when */patents/etl* or */pubmed/etl* is accessed.

## LUP Module for Semantic Search

The *Listen-Update-Predict* (LUP) implemented for the entity Search T4.5 task is the OSGi component responsible for listening, updating labelling prediction models by retrieving new annotations and sending them to the prediction algorithm. At prediction, when a query is typed by the user in the specific Firstswim Entity Search view, the list of entities is given by the LUP 4.5 as an RDF graph (the LUP Module responsible for the Entity Search task) through the PredictionHub.

**Overview of the component**

The LUP 4.5 is responsible for fetching the proper data for the learning task from the Annostore and provides a service that allows predictions on the labels of a document using the predict method below:

```
public String predict(HashMap<String,String> params);
```

The HashMap *params* must contain the following keys, which are the same as the parameters used to address the ECS search service:

1. *search:* the query typed by the user
2. *offset:* the offset of the ranked result you wish to apply
3. *maxFacets:* the maximum number of facets to display
4. *items:* the number of items we want to get back from the search

The returned value of this method produces a String object containing a serialized RDF-Graph which is detailed in the next section.

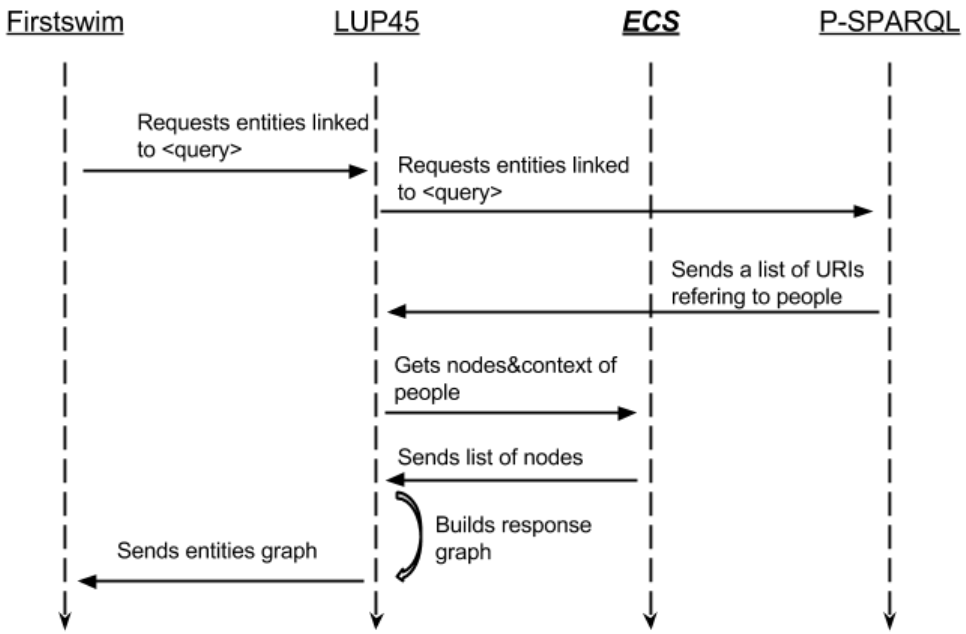


**Figure 9. Sequence diagram of the P-SPAQL engine in Fusepool**

This feature works following these five steps:

1. A query typed by the user is sent to the LUP45 bundle (via the PredictionHub component),
2. Given this query, the LUP45 bundle requests the Predictive Sparql engine on the OpenXerox server to get a list of people in relation with this query,
3. The OpenXerox server sends back a list of URIs to the LUP45 module,
4. The LUP45 builds a response graph using the Clerezza Content Graph and the list of people URIs,
5. Finally, the LUP45 sends this response graph back to Firstswim for display.

**Graph response**

The LUP45 sends back a ECS-like response graph oriented for people instead of documents. So we can get from the graph:

- Data about people:

```
<http://platform.fusepool.info/id/18b0a313-bd0c-40d3-b91e-84de3b9a72a8>
        a  <http://www.owl-ontologies.com/sumo.owl#Human> ,
<http://xmlns.com/foaf/0.1/Person> ;
        <http://www.w3.org/2000/01/rdf-schema#label>
                "AMAGASA TOSHIAKI" , "AMAGASA, TOSHIAKI" ;
        <http://schema.org/address>
<http://platform.fusepool.info/id/1ef8b03b-1241-442b-a22e-d2fa2eecc213>
        <http://www.patexpert.org/ontologies/pmo.owl#inventorOf>
                <http://platform.fusepool.info/doc/patent/EP-1008399-A1>
        <http://xmlns.com/foaf/0.1/name>
                "AMAGASA TOSHIAKI" , "AMAGASA, TOSHIAKI" .
```

- Data about the address of these people:

```
<http://platform.fusepool.info/id/a0194cf5-6f25-4f9f-a064-8b9d65e1caa6>
        a       <http://schema.org/PostalAddress> ;
        <http://schema.org/addressCountry>
                <http://platform.fusepool.info/code/country/DE> ;
        <http://schema.org/addressLocality>
                "D-41540 Dormagen" ;
        <http://schema.org/streetAddress>
                "Pfauenstrasse 51" .
```

- Data about documents (patents and/or publications) they've been contributing to, which is the same kind of data you would get back from the ECS.

This graph is built accessing the content store through the Clerezza *graphNodeProvider* component. We use the Clerezza *serializer* component to send the graph to Firstswim as a Turtle text file.

The usage of these web services is described in D5.2. An example of screenshot of the resulting GUI is shown on Figure 10. On this example, a query "ink jet printer has been made" and the user manually created labels associated to the documents.

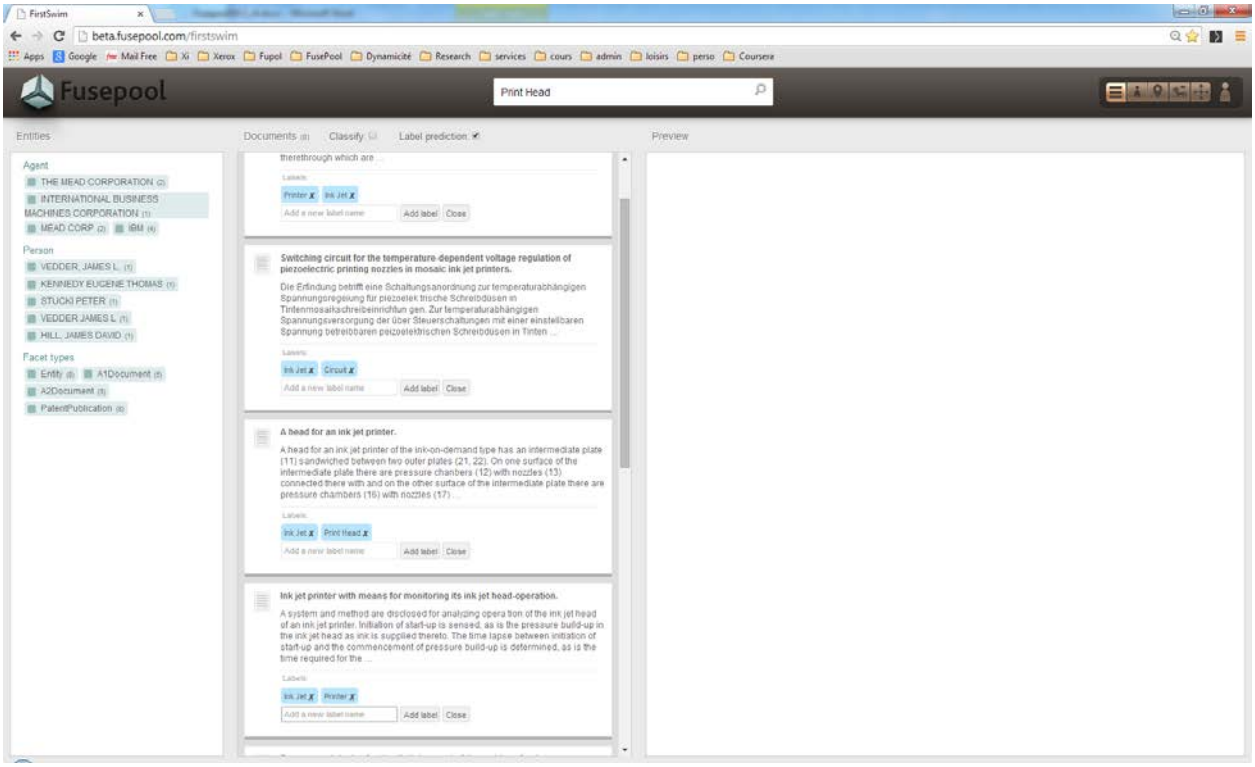


**Figure 10. Screenshot of the labeling interface enabled by the MTLS web services.**

# V. Conclusion

In this document, we describe the semantic search functionalities of the Fusepool platform. It represents a key innovative feature of the project.

On a scalability point of view, we showed a proof of concept for Latent Semantic Indexing using Lucene that could potentially further speedup retrieval, while keeping a high accuracy in the retrieval thank to the polysemy aspect provided by the factorization model.

For the user, it provides a seamless interface to the categorization of any entity type, not only documents. In the patent and publication examples provided by the Fusepool partners, we can efficiently create categories for people and institutions, but in the future, we can expect to easily develop the ability to categorize documents at a finer grain, for example by giving different categories to paragraphs, sentences or words.

The technical challenges we had to face we the ability to define a generic prediction framework using the Listen-Update-Predict (LUP) paradigm already used in previous deliverables. We developed a predictive query engine based on the re-interpretation of SPARQL queries and implemented a SPARQL parser that makes probabilistic computation. While very generic in principle, we applied this framework in the prediction of people categories within the Firstswim interface.

Finally, one the algorithmic point of view, we showed that tensor factorization is a powerful paradigm to create a predictive model of knowledge base that is sufficiently generic to be applied to many different domains. The implementation of the algorithm is based on highly optimized linear algebra packages that make the approach scalable to millions of entities.

While most of the results have not been widely tested by the final users, we believe that the innovative ideas introduced in this work open the door to many improvements in the future.

# VI. References

Bergman, M. (2005). *Untapped Assets: The $3 Trillion Value of U.S. Enterprise Documents.* BrightPlanet Corporation White Pape.

Berry, M., Dumais, S., & O'Brien, G. (1999). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Rev., Vol 37, No.4*, 335-362.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). An introduction to latent semantic analysis. *Journal of the American Society for Information Science*, 391-407.

Dumais, S. (2003). Data-driven approaches to information access. *Cognitive Science*, 491-524.

Grimes, S. (n.d.). *Unstructured Data and the 80 Percent Rule*. Retrieved from http://clarabridge.com/default.aspx?tabid=137&ModuleID=635&ArticleID=551

Hofmann, T. (1999). Probabilistic latent semantic indexing. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, 50-57.

Husbands, P., Simon, H., & Ding, C. (2000). On the Use of Singular Value Decomposition for Text Retrieval. *SIAM Computer Information Retrieval*, 145-156.

Kumar, C. A., & Srinivas, S. (2006). Latent Semantic Indexing Using Eigevalue Analysis for Efficient Information Retrieval . *International Journal of Applied Mathematics and Computer Science*, 551-558.

Nickel, M., Tresp, V., & Hans-Peter, K. (2011). A three-way model for collective learning on multi-relational data. *ICML*, (pp. 809--816).