Fusepool Linked Datapool for Technology Intelligence

Michael Kaschesky[1*], Reto Bachmann-Gmür[1], Guillaume Bouchard[5], Stephane Gamard[3], Anton Heijs[4], Michael Luggen[1], Tamás Prajczer[2], and Luigi Selmi[1]

[1] Bern University of Applied Sciences, E-Government Institute, Bern, Switzerland
{firstname.lastname}@bfh.ch
[2] Geox KFT, Budapest, Hungary
prajczer@geox.hu
[3]Searchbox SA, Lausanne, Switzerland
stephane.gamard@searchbox.com
[4]Treparel Information Solutions B.V., Delft, Netherlands
anton@treparel.com
[5]Xerox Research Europe, Machine-Learning Group, Meylan, France
guillaume.bouchard@xrce.xerox.com

Abstract. Fusepool applications for technology intelligence merge open data from millions of patents, publications, and research tenders making it available as keyword-searchable linked data. The open-source Fusepool data and software infrastructure separates the data, its processors, and the user interfaces from each other enabling users to transform and enrich their own data as linked data for searching and interlinking internal and external sources. Fusepool integrates many Apache projects and makes it easy for developers to build new apps using the ready-made OSGi Fusepool archetypes. The JavaScript client provides a semantic REST endpoint to be used alongside the SPARQL endpoint. Fusepool data undergoes a series of data extraction and refinement steps that remove ambiguity and make data reusable. As a result, the exploitation by app developers and the management of data becomes more efficient.

Keywords: technology intelligence, semantic search, linked data, fusepool

1 Introduction

Fusepool provides an open-source integrated data and software infrastructure that separates the data, its processors, and the user interfaces from each other. The Fusepool **end user applications** (e.g. PartnerMatch, PatentExplorer, FundingFinder, UserFeedback, PublicationExplorer, and ExpertMatch) integrate **large amounts of diverse and heterogeneous information sources** from different owners (e.g. EPO, PubMed, CORDIS) while preserving the **meaning of the different information sources** using RDF as the common data model and shared vocabularies (e.g. PatExpert, Bibo, FOAF. SIOC). Fusepool data undergoes a **series of data extraction and refinement steps** that remove ambiguity and make data reusable. As a result, exploitation by app developers and the management of data becomes more efficient.

2 System requirements specification and evaluation

The Fusepool team developed the system requirements and applications and their ongoing adaptation in close cooperation with end users and domain experts. Based on the living lab approach to software development involving 15 end user organizations, **rigorous evaluations** were and are done to **demonstrate and validate the results**. Because of their involvement, end users have confirmed the **clear commercial potential** of Fusepool applications (e.g. PartnerMatch, PatentExplorer, FundingFinder, UserFeedback, PublicationExplorer, and ExpertMatch).

2.1 Identification and definition of Fusepool user case studies and use cases

During the demos and presentations of Fusepool applications, the stakeholders interactively participated by asking clarification questions and providing their views and assessment. In addition, their responses in the applications for the Open Call, we evaluated all responses from the 18 shortlisted applicants, which together provided 61 user stories on the application scenarios and the questions they address.

The four application scenarios that were already defined during the 1st Fusepool User Workshop were defined as follows:

- **PartnerMatch**: How to find partners sharing similar/complementary traits?
- **PatentExplorer**: What is the trend and landscape of patents in my research field?
- **FundingFinder**: How to find and match call for tenders to my research interests?
- **UserFeedback**: How can input and feedback be used to optimize search results?
- **PublicationExplorer**: How to find and match papers to my research interests?
- **ExpertMatch**: How to find and match experts in a given field of expertise?

Fusepool application scenarios have in common that they find information that is best suited to address a specific user need. Hence, the use cases in each application scenario can broadly be divided into the following three groups.

- **Searching and retrieving information**: In the simplest scenario, the user only enters a search string describing the field of interest. The search engine uses the search index of all information available in the datapool to identify and rank the most relevant results based on a scoring algorithm.
- **Limiting retrieved entities or retrieving related information for a result**: The user may refine the search results in two ways or a combination thereof. By clicking on one of the types of *searched* entities, only this type will be searched thereby limiting the scope of *searched* content. Similarly, the user may click on one of the types of *retrieved* entities, thereby limiting the scope of *retrieved* information.
- **Providing feedback or annotations to search results**: The user may 'write' back to the datapool by adding personal annotations and customizations that can be shared among multiple users. For example, a user can select items from a search result list that are most relevant and trigger a classifier to calculate a classification score and re-rank the results accordingly. The classifier can be stored for later reuse.

Examples of real user stories

"Searching partners for fitting into a project or collaboration is essential, for our internal projects and for our SMEs. … Fusepool could help us to bring up new partners not included in our network. An example is the Knowledge and Workflow Management for Ports. We have been requested by one of our companies to help them to find technologies for increasing the efficiency in the value chain of Ports Management. We are using our network contacts and also the access to the main databases of Spanish Association of Science parks (SEIMED) in order to identify potential partners, technologies and also patents that could be related."

"FundingFinder and PartnerMatch will help our collaborators (and possibly other partners) in their daily business with the clients (researchers, SME and industry). … PartnerMatch: help client to build a project consortium, complements existing tools and network connections that we have utilized as part of our own core business (Enterprise Europe Network and Ideal-ist network partner searches for instance). FundingFinder and PartnerMatch are the most interesting and immediately applicable tools, as benchmarking for our own Opportunity-Finder results."

"To make the real world a testbed, we need to find the right technology for multi-source air quality sensors and analysis since the use case is different from traditional solutions. Fusepool might help targeting the right technology (patents), finding suitable partners, and joining related EU projects. … The PartnerMatch usecase gives those Chinese companies an easy portal to target the potential business partners. … We hope more than 20 companies from different sectors of China will join to use Fusepool."

"PartnerMatch is surely the most interesting because it closely overlaps with our efforts planned for the new member platform as described above. … [it] could support our high-tech firms developing their products further with the help of other partners and funding from external sources. It would enable them to invest in more riskier areas which they would not be able to fund themselves due to the larger risks involved (medium term research)."

"[We] want to make the newly developed diagnostic methods available to potential partners in the pharmaceutical industry. These would be companies or institutes that perform preclinical trials with animals for CNS drugs. … For example, when I have developed and validated a battery of tests as above, I want to offer this accurate selection of tests to the industries and for validation of CNS drugs (pre-clinical trials). For this I need a tool that can tell me which pharmaceutical group works on the pathology this battery of test targets and which CNS drugs they develop in relation to this/these pathologies identification of the company. Then I need to know who is the contact person for pre-clinical testing of the drugs developed in this company…"

3 Component integration and data storage

The Fusepool platform **scales in terms of integrated components** based Java OSGi.[1] Fusepool data sourcing and storage **scale in terms of integrated data** using common data formats (such as RDF) and shared domain-specific vocabularies (such as PatExpert for patents and Bibo for publications). Because of the reuse of shared vocabularies, interlinking to data resources in the Linked Data Cloud, potentially **reusing all related data currently published on the Semantic Web**. Data sourcing makes **use of dynamic data** as its being published at the source (e.g. PubMed for publications).

3.1 Fusepool platform and enhancers

The Fusepool platform supports supports the established Linked Data standards and best practices. It does so by relying on existing open source projects, notably Apache Clerezza and Apache Stanbol. Clerezza is a service platform based on OSGi (Open Services Gateway initiative), which provides a set of functionality for management of semantically linked data accessible through RESTful Web Services and in a secured way. Clerezza enables easy development of semantic web applications by providing tools to manipulate RDF data, create RESTful Web Services and Renderlets using ScalaServerPages. Stanbol enables the chaining of several semantic services, e.g. tag extraction/suggestion, text completion in search fields, 'smart' content workflows based on extracted entities, topics, etc. In addition, Apache Marmotta is investigated, which provides an open implementation of a Linked Data Platform that can be used, extended, and deployed easily by organizations wanting to publish Linked Data or build custom applications on Linked Data.

Each enhancer uses the input data in a different way in order to perform its task, but all share the same administrative interface, Java API and RESTful web service. A file sent to an enhancer through one of these two APIs will be mapped into an object, so called content item, to which all the enhancements will be attached in its metadata field as RDF data. A series of enhancers can be put in chain to form data pipelines. In this way the result of an enhancer can be used as an input to the following component.

Concerning component interoperability, while OSGi and adherence to good development practices ensures interoperability of components running within the same Java Virtual Machine (JVM) for interaction between applications running in different environment the interaction is done via HTTP and ideally via interfaces designed following the REST principles[2]. The Frontend components we contributed to Apache Stanbol foster the development of components that qualify as semantic and RESTful. This means that a client only needs a single service URI and understand the ontologies being able to fully use the provided services.

[1] http://en.wikipedia.org/wiki/OSGi

[2] REpresentational State Transfer (REST) is an architectural style, defined by Dr. Roy Fielding

[2] REpresentational State Transfer (REST) is an architectural style, defined by Dr. Roy Fielding in *Architectural Styles and the Design of Network-based Software Architectures*.

3.2 Fusepool content storage

Regarding interoperability of data resources, the Fusepool Platform implements the Enhanced Content Store (ECS), a new storage component,[3] which among other advantages provides better integration in the context of Linked Data than the original Stanbol Contenthub. The new content store enables uploading of documents so that both the document and their generated metadata can be dereferenced at persistent HTTP URIs as well as the integration of data from remote sites. This data is indexed locally and a copy of the triples is stored to allow fast queries but when users access one of described resources they are pointed to the original location. This design makes Fusepool a data service provider in the existing distributed linked data web and not a closed data silo. RDF offers the following improvements: linked data resources, structured and non-structured data accessible via a unified interface, access control and semantic REST APIs can be used.

3.3 Fusepool user authentication and authorization

The chosen approach leverages existing security mechanisms built into the Java Platform[4] and some libraries like Apache Clerezza[5]. The Java security model is based on a "sandbox" in which the code has limited and well defined access rights to resources both within and outside the Java Virtual Machine (JVM). When Java applications are run in secure mode at various points of their execution security checks might take place, execution only continues if the code is executed with sufficient permissions, otherwise a SecurityException is thrown.

The permissions system is very fine grained so that different operations can all have dedicated permissions. However it is also possible to have more coarse-grained high-level permissions and then have code sections which are executed as privileged executed without permission check. These patterns allow system administrators to more easily manage the permissions of users. For example if a user has the permissions to send emails the administrator doesn't need to additionally give the user the right to access the network, the high level permission "Send Email" implies more fine grained permissions like "Open Network Socket".

Consistently with the design paradigms of the platform the users are described in RDF. This security sensitive information is stored in a system graph. The system graph has itself the most restrictive access control settings. In this system graph users are mapped to roles and roles and users are matched to permission. The permissions are described using the standard syntax to describe them as it is used in Java Policy files[6], For describing the users and roles standard ontologies like FOAF and SIOC are used wherever possible.

[3] The Fusepool Enhanced Content Store is available here: https://github.com/fusepool/fusepool-ecs

[4] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html

[5] http://clerezza.apache.org

[6] http://docs.oracle.com/javase/6/docs/technotes/guides/security/PolicyFiles.html

4 Mapping data to RDF, entity extraction, and interlinking

Fusepool enhancers and bundles (e.g. NER, SMA, Interlinking, Smushing, KMX bundle) provide **functionality that goes beyond pure information retrieval**. NER provides domain-specific extraction and URI-generation of named entities detected within the corpus of text content while SMA provides a fast and scalable way to detect already known entities based on a dictionary of concepts and their URIs.

4.1 Automated transformation of data into RDF

Three RDFizers have been developed as Apache Stanbol enhancers to transform documents from XML or CSV to RDF. The work has been organized in two subtasks. Data mapping focuses on the transformations for all the files that were chosen as source: patents from the MAREC corpus, PubMed articles and FP7 calls. RDFizers focus on including such transformations into OSGi bundles in order to provide an easy usage of those transformations within the platform. The first two transformations were from XML to RDF/XML and have been developed as XSLT style sheets. The FP7 calls, originally provided as CSV files, have been transformed into RDF using SPARQL rules. Each subtask is described in the following paragraphs.

The design approach for URI/IRI patterns follows some of the best practices. In cases where the identifiers that could be easily algorithmically constructed from the data source or follow a common pattern, they were used directly in the pattern in order to create predictable (resource-friendly) and human-friendly IRIs; unique identifiers which were available directly in the documents e.g., patent identifiers, patent classifications, publication identifiers, funding call and topic identifiers. In other cases where the occurrences of *things or concepts* cannot be absolutely differentiated from one another, UUID values were generated and used in the IRI pattern.

4.2 Named entity recognition and disambiguation and co-reference resolution

Fusepool NER is a Java-based Apache Stanbol enhancer based on Stanford NER that produces an RDF output containing the extracted entities from the input text. The enhancer engine contains multiple NER instances, each instance is a separate module, and therefore each module has its own configuration page inside Stanbol Configuration Manager. It also means that different NER instances can be part of different enhancer chains. The enhancer consists of two main parts, initialization and production.

Fusepool SMA (String Matching Algorithm) is a Stanbol enhancer based on the Aho-Corasick SMA that produces an RDF output containing the extracted entities from the input text. SMA offers a more simple way for extracting information from texts. The enhancer engine contains multiple SMA instances, each instance is a separate module, and therefore each module has its own configuration page inside Stanbol Configuration Manager so that different SMA instances can be part of different enhancer chains. The enhancer consists of two main parts, initialization and production. The dictionary is stored in-memory in a hash table to make the lookup fast.

4.3 Information enrichment and interlinking

The result of an enhancement process for unstructured text or a transformation of a semi-structured document into RDF is a collection of disconnected RDF triples. In order to connect different representations of the same entity a reconciliation or interlinking task must be performed.

To start the interlinking some criteria must be defined for each entity type as to when entities of that type qualify to be interlinked. The descriptions to be compared must use the same or equivalent properties for which a mapping can be defined. The comparison of the properties values is based on different well-known similarity functions. The value of the comparison depends heavily on the quality of the data. If the computed value is above a certain threshold it is assumed that the two URIs represent the same entity. An identity relationship between two URIs is asserted using the *owl:sameAs* property. As a consequence of this new information all the statements about the entity found in the document can be asserted for the same entity in the knowledge base by a reasoner. In the Fusepool platform the interlinking process is provided by a component based on the Silk Link Discovery Framework. This component compares an entity's description, coming from enhancers or RDFizers, with entities stored in the ECS.

5 Keyword and faceted search for linked data and optimization

Fusepool applications (e.g. PartnerMatch, PatentExplorer, FundingFinder, UserFeedback, PublicationExplorer, and ExpertMatch) integrate the **user's contextual information** for rankings to **make results as accurate as possible** for the user. In fact, the ability of users to annotate content and results by adding new triples is a **novelty in applying semantic technology** to the social web.

5.1 Finding, evaluating, matching and integrating content

All data and relevant metadata are stored to an RDF graph with their literals and URIs indexed in Lucene for fast full-text search as well as *faceted search*. In contrast to Stanbol's ContentHub, the found entities are interlinked RDF resources so that it becomes trivial, for example, to display details of a found category. Because of the indexing at graph-level, existing RDF data can be added and benefit from the same indexing capabilities and from a unified access interface as uploaded and RDF-transformed content. This can be used to store relevant triples from existing sites on the linked data web. Following the REST principle, a client must only know the URI of the service, be able to parse an RDF format, and understand the ontology.

The *auto-suggestion* component creates a model of most likely queries by looking at phrases in the stored content and, when queried with either partial words of whole words, suggests a list of phrases which appear with the highest probability in the corpus. This leads not only to human readable suggestions, but guarantees the presence of documents when a phrase is selected. It gives the user a intuitive feeling of what type of information is available in the system, making their foraging activities easier.

5.2 Optimizing results based on user feedback

The components dealing with unstructured content are sensitive to ambiguities, noise, and other forms of mistakes in prediction. For example, a simple NER implementation is often based on a dictionary lookup. That is, a text is examined one word at a time and if a word appears in a specific type-based dictionary, the word is tagged with that type. The problem comes when words can appear in multiple dictionaries. The word "cancer" appears in an "astrological sign" dictionary but also in a "medical disease" dictionary. If the system has erroneously tagged the word with the astrological meaning, this mistake can be manually corrected. However, this kind of error is common and we do not want a user to have to manually correct errors that are related.

KMX is a client server architecture made available in Fusepool with specific features specifically for advanced users like professional patent searchers. It is implemented via a web service as a RESTful API to be used from within the Apache Stanbol environment. The web service provides REST endpoints for the following tasks: Managing datasets (Creating, Adding); Workspace (Creating from a dataset, Labelling for classifier model, Creating stop lists); Classification (Training/Applying SVM models); and Clustering (e.g. Landscape). The process of optimizing results based on user feedback is detailed below.

6 Graphical user interfaces and visual analytics

For a **range of mobile and desktop devices**, Fusepool provides an **attractive and functional web user interface** through its applications (e.g. PartnerMatch, PatentExplorer, FundingFinder, UserFeedback, PublicationExplorer, and ExpertMatch).

6.1 Visualization concept and dashboard framework

The Big Data vision is popularly defined using the *Three V's*: Volume, Velocity and Variety[7]. However, big variety of data makes it difficult to provide smart user interfaces for the data. A general approach provides a user interface for about any kind of data while a very specific user interface targets a specific kind of data. Both approaches are not satisfactory: in the generalized UI valuable information is missing or the user is overwhelmed, while a specific interface might not be used for other kinds of data and is not scalable to other requirements.

To solve this problem, a hybrid approach is used: The Fusepool dashboard framework provides the basis of the graphical user interfaces for user interactions. For example, users can execute search functions in the dashboard and drilldown and refine results. Gradually, the dashboard becomes more adaptive and dynamic, first by adapting the graphical user interfaces and information visualizations to multiple devices (PC, tablet, smartphone) and second, the graphical user interfaces and information visualizations become adaptive to the information, i.e. the semantics of the data itself.

[7] As defined by Gartner at http://www.gartner.com/newsroom/id/1731916

Designing and prototyping the graphical user interfaces (GUI) involved an iterative process from discussions with project members, end users, designers and programmers over physical mock-ups to quickly implemented functionalities using the Enyo framework[8].

One of the core tenets of Fusepool is to separate the data from the GUI and visualizations. The dashboard is "intelligent" meaning that its layouts and designs are adaptive based on device characteristics and user profiling and customizations. The traditional approach is that application developers decide what data should be shown creating a GUI based on software architecture patterns such as the model-view-controller. However, such a system would be tightly coupled and changing the data usually requires redesigning the GUI accordingly.

6.2 Intelligent dashboard implementation

The GUI is built on Enyo with different modes for PC, tablet, and smartphone. The final implementation contains two separate interfaces: a large screen version and a mobile version. The data is fetched in JSON from a specialized server for search query processing and querying of the RDF data. The detailed data of the entries is than fetched directly from a SPARQL endpoint to the client. The RDF data is parsed using the rdfquery library to handle it via the GUI.

The GUI starts out clean and uncluttered with just a search interface. When the user enters a query term, auto-suggestion provides a list of likely phrases that may complete the query. After every typing in the input field, an Ajax request is sent to the backend to get a list of likely phrases for the query term. Pressing 'Enter' or clicking the search symbol, the GUI sends the request to the Fusepool platform.

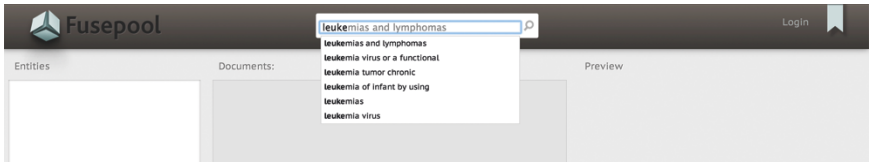

Figure 1: Search with auto-suggestion for query terms

The search results are displayed in the middle pane of the GUI. On the left side, the entities are displayed under their types (facets). The GUI groups the entities by types and sorts them according to the number of entities for each type – the type with the most entities is displayed at the top. To avoid cluttering the GUI, each entity is shown only under one type. The identified entities in the retrieved list of files are displayed on the left pane. If the user clicks on the checkbox left to the entity's label, then the GUI starts a new search where only files with that entity are retrieved. The list of retrieved files updates instantly.

[8] Enyo is a JavaScript app framework to build native-quality HTML5 apps that run everywhere [http://enyojs.com]

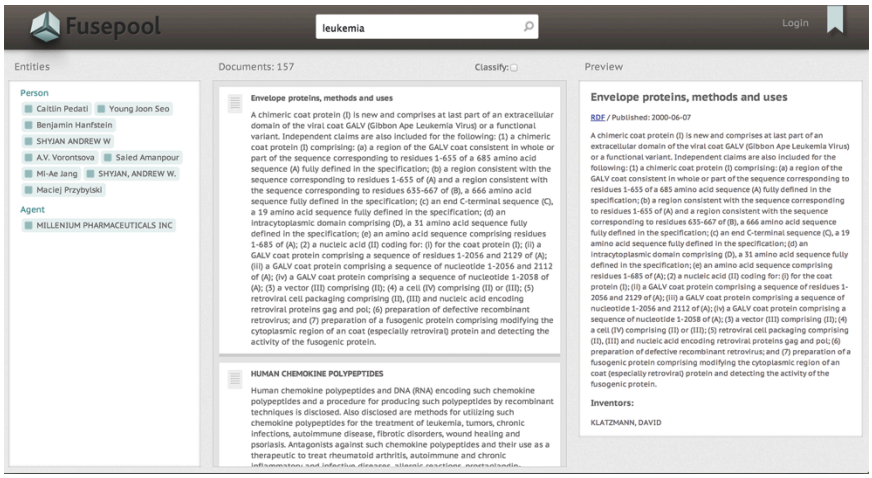

Figure 2: Display of search results (middle), detail view (right), and entities as

In order to better match search results to a user, the user can create a text classifier to be applied to those or similar queries in the future. To train a text classifier, the user must first train the classifier selecting a sufficient number of files as positive (relevant) and negative (not relevant). To start training, the checkbox next to 'Classify' is checked, prompting the GUI to display interfaces for the training.

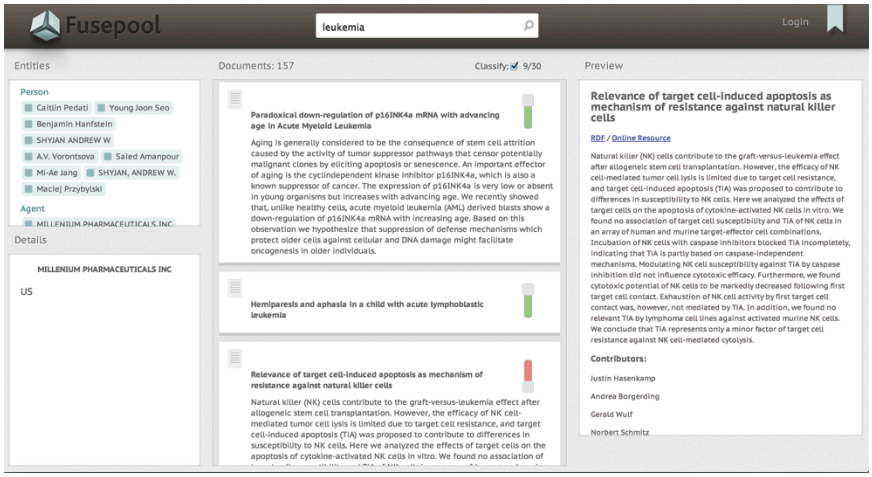

Figure 3: Training a text classifier by selecting positives and negatives