# CAPLAN: An Accessible, Flexible and Scalable Semantification Architecture (Project Description)

Sebastian Furth[1], Volker Belli[1], Alexander Legler[1], Albrecht Striffler[1], and Joachim Baumeister[1,2]

[1]denkbares GmbH, Friedrich-Bergius-Ring 15, 97076 Würzburg, Germany
[2]University of Würzburg, Institute of Computer Science, Am Hubland, 97074 Würzburg, Germany
{sebastian.furth,volker.belli,joachim.baumeister}@denkbares.com

**Abstract.** The popularity of semantic information systems requires more data to be semantically prepared. However, the subsequent semantification process is still reserved for experts in Natural Language Processing. In this paper we define requirements for a state-of-the-art semantification architecture. Additionally we present a concept for a new semantification architecture meeting these requirements. Key strengths of the presented concepts are accessibility for non-experts, scalability and flexibility.

**Keywords:** Semantification,Information Management Architecture,Knowledge Management

## 1 Introduction

Semantic Search [7] emerged as the new system paradigma for enterprise information systems. In contrast to traditional information systems Semantic Search exploits ontologies during the retrieval process. The search performance usually outperforms traditional text based retrieval engines. However the underlying semantic search engines require resources to be semantically prepared. The semantic preparation [5] / semantification process of such resources typically comprises the partition of resources to reasonable segments, so called information units, and the subsequent semantic annotation with concepts from an ontology. The process is typically realized as a sequence of process steps.

The popularity of Semantic Information Systems leads to an increased need for migrating existing resources to semantic representations. However, existing implementations of the semantification process typically require a decent amount of knowledge in Text Analytics / Natural Language Processing and are thus hardly accessible for non-expert users. Additionally, implementations usually lack scalability and are thus not well prepared for processing large amounts of data. In most cases they are also inflexible with respect to the underlying data model and are thus hardly customizable to specific project needs.

In this paper we present a concept for a novel semantification architecture that is part of the ongoing research project APOSTL. The architecture is powered by a flexible state-of-the-art data model that is well prepared for the usage in scalable high performance environments. The easy management of project resources, import of existing data as well as assessment and review components open the semantification process for non-experts.

The remainder of the paper is structured as follows: In Section 2 we first describe requirements for a state-of-the-art semantification architecture. In Section 3 we explain required components and give some remarks to future implementations. Related work is briefly considered in Section 4. We conclude in Section 5.

## 2 Requirements

The overall requirements to the architecture are accessability for non-expert users, scalability to large-scale data sets and flexibility for new project requirements. In the following we break down these requirements.

### 2.1 Accessability

The increasing amount of semantification projects requires that semantification processes are accessible for non-experts (wrt. to Text Analytics/Natural Language Processing). This requires that the architecture is able to **hide the complexity** of underlying NLP processes. Users without expert knowledge in Natural Language Processing should be able to configure the semantification process on an abstract level, without having to know specific details of underlying approaches.

The opening of the semantification process to non-expert users requires that the architecture provides **documentation** for each of the underlying process steps. The documentation for each process step has to state clearly what data in which format is required as input and which results can then be derived from this data as output.

The generated data should be provided with **provenance and versioning information** that states clearly how (which method and parametrization) and when the data has been produced. The availability of such information facilitates the reproducibility of results and the comparison of parameter configurations.

The architecture should also provide ways to **examine generated results** on a high level. Therefore, the data visualization techniques should be a vital element in the architecture to open the assessment of results to a wide user range. Additionally, interactive review tools should allow the users to easily correct generated results.

Another aspect of accessability affects the representation of the underlying data. Due to their subsequent usage in semantic applications all (intermediate) results should have a **semantic representation**, i.e. all data elements should at least be identifiable using a URI and provide type information.

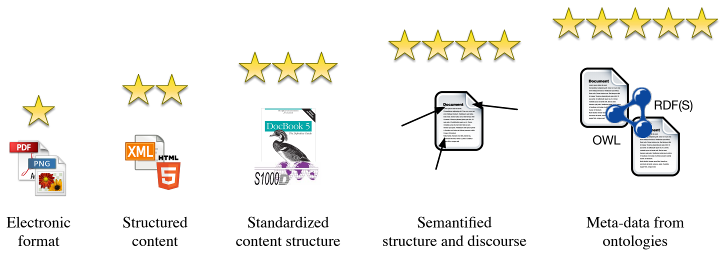
## 2.2 Scalability

Scalability has a two-fold meaning in the context of semantification architectures. It is primarily concerned with the support of **large scale data processing** (Big Data), i.e. the architecture should be prepared to be employed in high performance environments for high throughputs. This requires that underlying algorithms are available for Big Data processing frameworks like Apache Spark [10] and the underlying data model supports distributed data storages like Hadoop's HDFS [11].

However, scalability in this context is also concerned with the aspect that a wide range of users should be able to use the semantification architecture. Therefore, the architecture should be realized as **Business Process as a Service**. A business process as a service is typically realized as a cloud service. In the context of a semantification architecture this means that the whole semantification process is available as web application or API.

## 2.3 Flexibility

A semantification process typically comprises a series of complex operations that successively prepare a resource for the usage in a semantic information system. However, in some cases some of the operations are not necessary, because data is already prepared to a certain extend (cf. Figure 1). Therefore, users should be able to enter the semantification process at an **arbitrary process step** if they can provide data in the necessary format.



PDF
PNG
XML
HTML
DocBook 5
S1000D
Document
RDF(S)
OWL
Electronic format
Structured content
Standardized content structure
Semantified structure and discourse
Meta-data from ontologies

**Fig. 1.** Maturity schema for documents in the semantification process.

Sometimes the semantificaiton process must not necessarily be completed, e.g. because intermediate results are sufficient for specific application scenarios. Typical examples include specialized Information Extraction tasks that operate on semantically represented document structures. Hence, the architecture should allow to **query and export intermediate results**.

Although the process steps of semantification processes are usually similar in various application scenarios it might be necessary to parametrize, extend or adapt the process to new process requirements. Typical scenarios include the existence of a previously unknown source format or new approaches/parameter configurations for specific process steps like segmentation, term matching or subject indexing. Thus, the architecture shall be **extensible**, such that new process steps or variants of existing process steps can easily be integrated. The extensibility should also be reflected in the data model.

## 3 Architecture

In the following we present an architecture facilitating the semantification of resources under the requirements stated in Section 2. Therefore, we first introduce the key components of the architecture and then close the section with some remarks regarding future implementations.

### 3.1 Components

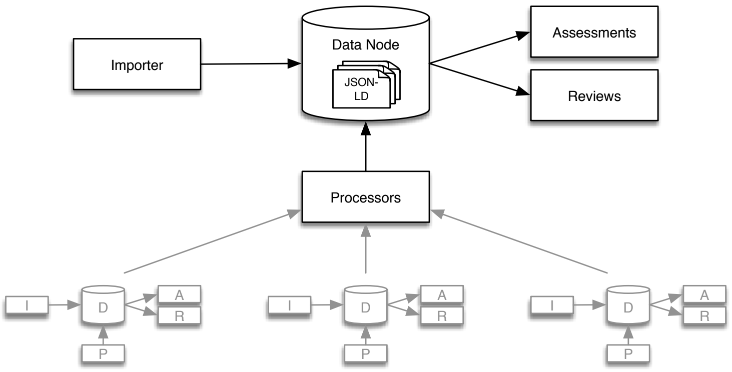


**Fig. 2.** Key components of semantification architecture.

Refering to the requirements in Section 2, a semantification architecture should be accessible, scalable and highly flexible. The flexibility mainly demands for a high extensibility and standardized import, export and processing functionality in all process steps while accessability is concerned with hiding complexity from non-experts, providing easy-to-use assessment and reviewing functionalities and standardized data representations. Thus, we propose an architecture (see Figure 2) that is composed of interweaved modules, that are represented as quintuples $Q = \{D, I, P, A, R\}$, with:

- **Data Nodes** $D$: Contain the data and a data description for the process step, e.g. a description of document structures and instance data for concrete documents.

- **Importers** $I$: Provide and document import functionalities for data nodes, i.e. describe possible import formats and handle the import of data nodes from raw/source data. Also creates provencance information for the imported data.
- **Processors** $P$: Process data nodes in order to produce new or update existing data nodes respectively. Also creates provencance information for the generated/updated data.
- **Assessments** $A$: Provide possibilities/metrics/visualizations to assess a set of data nodes.
- **Reviews** $R$: Allow manually changing/reviewing existing data nodes.

All elements of the quintuple except the data nodes are optional. A semantification system can be built by combining multiple modules to a complete process, where each module encapsulates specialized functionality for a certain process step.

The interconnection of the encapsulated functionalities is realized through the data nodes. All data nodes are stored in a common schema-less data base (NoSQL) and are from there accessible from all modules. This way, the output of one module can be used as data source from another module which itself can produce new data nodes and so on. Additionally the usage of a schema-less NoSQL data base ensures the extensibility of a system, as new data can be stored without constraints.

The interconnection of modules in a semantification system is explained by the example of segmentation, term matching and subject indexing. Therefore, we assume that we have three modules encapsulating the aforementioned functionalities. Then the procedure is as follows:

1. **Segmentation:** A importer imports raw documents and stores them as data nodes (when appropriate using references to original sources).
2. **Segmentation:** A processor partitions the raw documents to segments and stores them as data nodes.
3. **Term Matching:** A term matching processor configured with a list of relevant terms scans the stored segment data nodes for term occurrences. Discovered occurrences are stored as new/complementary data nodes.
4. **Subject Indexing:** A subject indexing processor accesses the segment data nodes and the corresponding term match data nodes. Based on the information it determines topics for the segments and stores them as new/complementary data.
5. **Subject Indexing:** An assessment component visualizes the subject indexing result, e.g. highlights segments with many or few subject annotations.
6. **Subject Indexing:** Based on the assessment, the parametrization of step 4 may be revised and step 4 repeated. With stored provenance information multiple outcomes can be compared and the most appropriate one selected.
7. **Subject Indexing:** A review component allows to edit subject annotations, e.g. remove unnecessary or add missing subjects respectively.

The (intermediate) results, namely segments, term matches and annotated subjects can then be exported for subsequent usage in other systems.

### 3.2 Implementation Remarks

The implementation of the proposed architecture or rather the corresponding framework has not yet started. However, we have already defined some parameters specifying the subsequent implementation. These parameters affect the data model, the graphical user interface and the module mechanism.

**Data Model** The complete architecture builds upon a very flexible schema-less data model. The data model will be implemented as document-oriented NoSQL data base, where documents are the basic storage entity. We require JSON-LD [12] as storage format, which is standardized, light-weight, well-supported in common data base systems and allows to use explicit semantics. The availability of JSON-LD also allows to export (intermediate) results as standardized ontologies [8,14]. Furthermore, JSON(-LD) is compatible with common high performance data bases that work upon Apache Hadoop, e.g. MapR-DB. Importers $I$ and processors $P$ has to enhance the JSON-LD documents with provenance information from the PROV-O [13] ontology.

**Module Mechanism** The architecture is based upon the idea that a semantification system can be composed of modules that encapsulate specialized functionality. Besides a description of the data nodes (if appropriate as JSON-LD context), a module can define importers $I$, processors $P$, assessments $A$ and reviews $R$. For the integration in the framework each of these components must provide specific information. Additionally, each component might define additional parameters that are necessary for configuration. Therefore, we plan to use a standardized plugin framework like OSGi [2].

Considering the scalability requirements modules should also report whether they are capable of running in high performance environments. Therefore, modules should express there high performance capability in their plugin definitions. If they claim to be high performance capable, we require them realize their functionality using a high performance computing framework like Apache Spark [10] or Apache Flink [1].

**Graphical User Interface (GUI) and API** As one requirement is a high accessibility for non-experts the framework will have a standardized graphical user interface. The graphical user interface shall guide users through existing semantification processes and allow for the creation of new/customized processes. Therefore, some components of the modules like importers or processors will be presented in a standardized way to allow the configuration by the user. Other components like assessments or reviews require a specialized user interface. Hence, these components must also provide user interface definitions as part of a module. The functionality that is accessible through the graphical user interface shall also be available as API to facilitate the process or module integration in other applications.

### 3.3 Requirement Tracing

In the following we give a brief requirement tracing, i.e. which requirement is realized by which component.

#### Accessability

- **Hide Complexity:** Importers $I$ and Processors $P$ allow for the import and processing of data in a documented format.
- **Documentation:** Importers $I$ provide documentation of importable data formats.
- **Provenance and Versioning:** Provencance and Versioning information are stored along with the data nodes in the common data base.
- **Examine Results:** Assessments $A$ and Reviews $R$ allow for the easy evaluation and review of results.
- **Semantic Representation:** All (intermediate) results are stored as JSON-LD documents with an explicit semantic.

#### Scalability

- **Large Scale Data Processing:** Module functionality can be implemented using high performance computing frameworks.
- **Business Process as a Service:** The framework will provide a standardized graphical user interface and an API.

#### Flexibility

- **Enter process at arbitrary steps:** Each module can have importers that allow the direct import of the required data.
- **Export (intermediate) results:** The results of each processing step can be exported as standardized ontology.
- **Extensibility:** The architecture allows for the easy extension through a module mechanism that will be realized using a plugin framework.

## 4 Related Work

To the best of our knowledge we are not aware of a framework that meets the requirements stated in Section 2 for a accessible, flexible and scalable semantification architecture. However, there are extensible frameworks for Natural Language Processing/Text Analytics tasks. Prominent examples are Apache UIMA [9] or GATE [4]. However, they usually need expert knowledge to be employed and come with a couple of shortcomings, cf. Bank et al. [3] for details. The idea of building specialized applications from standardized modules is not new, cf. for example Gu et al. [6].

## 5 Conclusion

In this paper we described early work from the ongoing research project CAPLAN. We presented requirements for a state-of-the-art semantification architecture. The requirements can be summarized with accessibility, scalability and flexibility. We then presented a novel semantification architecture that is composed of specialized modules that are interconnected through a very flexible and standardized data model based on JSON-LD. We showed that our architecture meets all the requirements and briefly named existing alternatives and their shortcomings.

Future directions include a further refinement of the presented architecture. Subsequently the concept will be realized in a prototypical implementation. The implementation will comprise the framework as well as sample modules for specialized semantification use cases.

## Acknowledgments

The work described in this paper is supported by the Bundesministerium für Wirtschaft und Energie (BMWi) under the grant ZIM ZF4172701 "APOSTL - Accessible Performant Ontology Supported Text Learning".

## References

1. Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J.C., Hueske, F., Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., et al.: The stratosphere platform for big data analytics. The VLDB Journal 23(6), 939–964 (2014)
2. Alliance, O.: Osgi Service Platform, Release 3. IOS Press, Inc. (2003)
3. Bank, M., Schierle, M.: A survey of text mining architectures and the uima standard. In: LREC. pp. 3479–3486 (2012)
4. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: an Architecture for Development of Robust HLT Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL) (2002)
5. Furth, S., Baumeister, J.: Semantification of Large Corpora of Technical Documentation. IGI Global (2016), http://www.igi-global.com/book/enterprise-big-data-engineering-analytics/145468
6. Gu, T., Pung, H.K., Zhang, D.Q.: Toward an osgi-based infrastructure for context-aware applications. IEEE Pervasive Computing 3(4), 66–74 (2004)
7. Guha, R., McCool, R., Miller, E.: Semantic search. In: Proceedings of the 12th international conference on World Wide Web. pp. 700–709. ACM (2003)
8. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-primer/
9. Lally, A., Verspoor, K., Nyberg, E.: Unstructured Information Management Architecture (UIMA) Version 1.0 (March 2009), http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html
10. Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: Machine learning in apache spark. JMLR 17(34), 1–7 (2016)

11. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). pp. 1–10. IEEE (2010)
12. Sporny, M., Kellogg, G., Lanthaler, M., Group, W.R.W., et al.: Json-ld 1.0: a json-based serialization for linked data. W3C Recommendation 16 (2014)
13. W3C: PROV-O: The PROV Ontology: http://www.w3.org/TR/prov-o (April 2013)
14. W3C: RDF Schema 1.1 – W3C Recommendation. http://www.w3.org/TR/rdf-schema (February 2014)