Towards Cognitive Pipelines of Medical Assistance Algorithms

Patrick Philipp[1], Darko Katic[2], Maria Maleshkova[1], Stefanie Speidel[2], Achim Rettinger[1], Benedikt Kämpgen[1], Sascha Zelzer[4], Anna-Laura Wekerle[3], Rudi Studer[1], Rüdiger Dillmann[2], Hannes Kenngott[3] and Beat Müller[3]

[1] Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
patrick.philipp@kit.edu, maria.maleshkova@students.kit.edu, achim.rettinger@kit.edu, benedikt.kämpgen@kit.edu, rudi.studer@kit.edu
[2] HIS, Karlsruhe Institute of Technology, Karlsruhe, Germany
darko.katic@kit.edu, stefanie.speidel@kit.edu, ruediger.dillmann@kit.edu
[3] Klinik für Allgemein-, Viszeral- und Transplantationschirurgie, University of Heidelberg, Germany
anna-laura.wekerle@med.uni-heidelberg.de, hannes.kenngott@med.uni-heidelberg.de, beat.mueller@med.uni-heidelberg.de
[4] Division of Medical and Biological Informatics, German Cancer Research Center (DKFZ), Germany
s.zelzer@dkfz-deutschland.de

Abstract. *Purpose:* Assistance algorithms for medical tasks have great potential to support physicians with their daily work. However, medicine is also one of the most demanding domains for computer based support systems, since medical assistance tasks are complex and the practical experience of the physician is crucial. Recent developments in the area of cognitive computing appear to be well suited to tackle medicine as an application domain.
Methods: We propose a system based on the idea of cognitive computing and consisting of auto-configurable medical assistance algorithms and their self-adapting combination. The system enables automatic execution of new algorithms, if they are made available as *Medical Linked APIs* and are registered in a central semantic repository through a rule-based engine. Learning components can be added to the system to optimize the results when numerous *Medical Linked APIs* are available for the same task. Our prototypical implementation is applied to surgical phase recognition based on sensor data and image progressing algorithms for tumour progression mappings.
Results: Our results suggest that such assistance algorithms can automatically be configured in execution pipelines, candidate results can be automatically scored and combined and the system can learn from experience. Furthermore, our evaluation shows that the *Medical Linked APIs* are providing the correct results as they did for local execution and run in a reasonable amount of time.
Conclusion: The proposed solution is applicable to a variety of medical use cases and effectively supports the automated and self-adaptive configuration of cognitive pipelines based on medical interpretation algorithms.

1 Introduction

Assistance algorithms for medical tasks have great potential to support physicians with their daily work. This includes image processing, simulations and interpreting sensor data, rule-based treatment recommendations and many more. However, medicine is also one of the most demanding domain for computer based support systems. This has diverse reasons, two important ones are:

(a) Since medical assistance tasks are extremely complex there is not one single algorithm which can solve a complex task alone. Typically, a combination of heterogeneous methods from probabilistic reasoning, knowledge based inference and information retrieval is needed. In addition, those methods need to be executed both, in sequence and in parallel, producing a list of candidate results.
(b) Medicine is not an exact science, so experience of the physician is crucial. Much of the knowledge can not be formalized easily, but is gained over years of hands-on-experience.

A very recent development in research that tackles this sort of complex tasks is cognitive computing, most known by IBM's Watson system [4]. While the initial application domain of Watson was general knowledge (the Jeopardy! Challenge) [4] IBM turned to medicine as a promising second application domain. Just recently IBM presented ongoing work on its Oncology Expert Advisor [2]. While the original Watson system is very much focused on Q&A over textual content, the principles of cognitive computing can be applied to many diverse medical assistance tasks.

We propose a reactive approach based on the idea of cognitive computing to execute algorithms annotated with ontological concepts based on the Resource Description Framework (RDF). Whenever possible, a learning component combines the results of different algorithms for a single task. We apply our methods to two medical use cases - surgical phase recognition based on sensor data and image progressing algorithms for tumour progression mappings.

Our prototypical implementation, suggests that when provided with real world data,

(i) assistance algorithms can be described and executed in an automatic fashion based on the context provided as input, instead of hand-engineered execution pipelines.
(ii) several alternative paths can be executed in parallel and generate a set of candidate outputs.
(iii) candidate outputs can be automatically scored and selected by evidence.

In this paper we call such auto-configurable medical assistance algorithms Cognitive Apps and their self-adapting combination and execution a Cognitive Pipeline.

The remainder of this paper is structured as follows: In the next section we will discuss related work, before introducing our two application scenarios. In

Sec. 4 the components of our software system are described. In Sec. 5 and 6 initial empirical results in our application scenarios are presented. The paper concludes with a discussion and future work.

2 Related Work

There is an ongoing research interest in so-called 'workflow systems' that enable describing and executing algorithms of different kinds. The work centered around semantic workflows [7] aims to enable the automatic composition of components in large-scale distributed environments. The combination of individual algorithms is supported through generic semantic descriptions also enabling ensembles of learners to be formulated. This requires stipulating necessary conditions and constraints of how to combine the algorithms. The workflow system is able to match components and data sources and provides for high degrees of automation when processing user requests.

Taverna [15] is a another scientific workflow system supporting process prototyping by creating generic service interfaces and thus easing the integration of new components. Semantic descriptions are being used to better capture the view of the scientists. Taverna is able to integrate data from distributed sources and automate the workflow creation process for users. A graphical workflow workbench can be used by scientific staff to manage their processes, hiding the complexities of the underlying architecture.

In [23] abstract workflows are being created as domain models formalized using the Web Ontology Language (OWL) to enable dynamic instantiation of real processes. These models can the be automatically converted into more specific workflows resulting in OWL individuals. The semantics of the data are well represented at any point during the workflow. The components can be reused by using them in another context or process and these abstract representations can be easily shared across the Web through OWL classes.

In contrast to previous approaches, our work benefits from lightweight semantics, minimizing the efforts required for describing the algorithms and the used data. Furthermore, it represents a novel and innovative contribution, since it combines semantic technologies with learning approaches. Currently, we are not focussing on GUIs. Instead, we aim to provide the users with transparency for what the system is doing, i.e. based on which assumptions it chose the algorithms and which evidence it used. As the above approaches, we also leverage semantic descriptions but use a combination of OWL and the Resource Description Framework (RDF) as basis, which makes our system considerably different in designing and providing the algorithms. In addition, none of the approaches employ a knowledge base storing performance results and trying to reuse this evidence in order to optimize results by learning.

3 Scenarios for the Self-Adaptable Algorithms

The work present here has be applied within the context of the cognition-guided surgery project SFB/Transregio 125[1] that aims to develop assistant systems for surgeons. In particular, empirical knowledge, facts and patient data are to be combined to identify and characterize the current medical situation and its needs, and eventually perform appropriate actions. Our system has been applied in the context of two use cases – *Surgical Phase Recognition* and *Tumour Progression Mapping*, which are described in detail below.

3.1 Surgical Phase Recognition

Surgical phase recognition and information filtering is in the focus of active research in medical information science [3], [11]. Surgeons today are faced with a huge variety of intraoperative information sources. Preoperative planning, CT/MRI images, vital signs of the patient and various device states are available at any point during the surgery. The problem is that only a small fraction of the available information is actually relevant in a given situation. Showing all information at once is likely to do more harm than good as the data will outpace human ability to process it [9]. In order to make full use of computerized surgery, we need methods to infer the current phase of the surgery and to thus automatically filter information and only present the currently relevant bits to the surgeon. Technically, phase recognition is done by analyzing situation features called activities [14], [12]. Activities are triples consisting of the used instrument, the performed action and the corresponding anatomical structure, e.g. <Scalpel, cut, Gallbladder>. We define phases as sequences of activities, in which the surgeon needs a specific set of information. The aim of surgical phase recognition is to analyze the currently occurring activities and infer the current phase. Intraoperativey , these activities are recognized using sensor analysis techniques [18]. For evaluation and testing, we use manually annotated videos of surgeries, where clinical experts provide the ground truth, on which activities occurred during the course of the surgery. The annotations were created using the SWAN-Suite software [13]. We are applying our approach to *SWRL Phase Recognition* introduced in [10] and a *Learning-based Phase Recognition* algorithm, which utilizes training samples (i.e. annotated surgeries with activities as mentioned before).

3.2 Tumour Progression Mapping

Tumour Progression Mapping (TPM) is an approach to visualize the timely progression of brain tumours for radiologists. The process of generating a TPM produces numerous images over time exhibiting different characteristics. Radiologists want to see the development of the glioblastoma since the last surgery but this prerequisites tedious and complex tasks. Reasons are that glioblastoma grow

[1] `http://www.cognitionguidedsurgery.de`

irregularly and assessments are being conducted by visually comparing scans in different point of time and different shifts. The lack of image registrations impair these problems. The transformation of the TPM approach applied to the follow-up appraisal of glioblastoma was published in [6] and [16] and will be integrated in this paper to emphasize the configuration capabilities of our system.

The workflow for tumour progression mapping is illustrated in figure 1. The images are stored in a PACS and converted into a common format (NRRD). A mask for the brain region is being created in the next step, ensuring that subsequent tasks are not influenced by bones or other structures. All prevalent brain images of a patient are then being spatially registered. The following normalization task adapts the intensities of MRI scans yielding similar values for similar tissue types. If additional annotations for a patient are prevalent, the normalization becomes more robust. The TPM can now be created. Optional additional steps are automatic tumour segmentation and subsequent integration into the map.

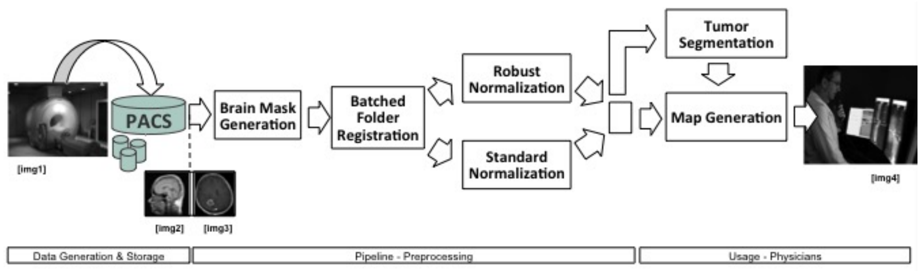

Fig. 1. Data-driven Workflow of TPM Image Processing Algorithms [16]

4 System Components

We enable medical interpretation algorithms such as surgical phase predictors or image preprocessors to automatically run when needed in potential ad-hoc workflows with optimal set ups. This is realized by implementing the following components:

1. A **Knowledge Base** with information about use cases, algorithms and data.
2. **Cognitive Apps** – algorithms are annotated with semantic metadata and made available on the Web. The algorithms access the necessary input data from the knowledge base and feed back the results.
3. An **Execution Engine** which automatically finds, initializes and runs the algorithms, based on the information stored in the knowledge base.
4. An **Evidence-based Learning** component to return the optimal result for the task at hand, given evidence in the knowledge base.

We use methods developed for the Semantic Web to create semantic annotations for algorithms, and the data they are consuming and producing. The Resource Description Framework (RDF) is being used to to create lightweight algorithm and data annotations. Concepts and instances, which are created, are integrated following the Linked Data principles[2] by persisting resources as URIs in the Web and creating links between them. Communication between components is enabled by using web technologies such as HTTP methods.

By combining these components we enable a cognitive pipeline which does not require manually defined procedures and improves over time due to consistently feeding the knowledge base. Cognitive Apps are semantically annotated interpretation algorithms which are made available by combining Linked Data concepts and web services.

4.1 Knowledge Base

In the following, we describe the system components in more detail.

Semantic MediaWiki A Semantic MediaWiki (SMW) is used to enable semi-structured annotations of information from both the medical and technical worlds, which are then instantly available as RDF under persistent URIs. The SMW enables domain experts to semantically annotate their use cases, algorithms and data to formally define their interaction. The central data taxonomies and a generic templates for describing algorithms are constantly manually updated based on new domain knowledge. Domain experts thus do not have to learn to modell OWL or RDF ontologies to create the semantic descriptions needed for self-adaptive interpretation algorithms.

Semantic Patient Data Store An instance of the XNAT[3] platform – an open source imaging informatics software platform – offers the possibility to store instance data for different medical departments across their boundaries. We developed a RDF conversion for the platform to be able to publish all patient-related data with common vocabulary. This enabled us to enrich the prevalent information structure of XNAT with newly defined predicates. One can either upload pure RDF with links to patients or upload non-RDF files and link them to centrally defined concepts. Ideally, these semantic annotations suffice to characterize the files completely.

4.2 Cognitive Apps

Semantic Algorithm Descriptions We developed a generic template for capturing the necessary semantic annotations in order to describe the interpretation algorithms. It consists of functional and non-functional requirements. Each functional and non-functional requirement was modelled in the SMW and can be used as a semantic annotation.

Non-functional requirements comprise the following information:

[2] `http://linkeddata.org`

[3] http://www.xnat.org

- *Name:* A unique name or identifier for the algorithm within the project.
- *Contributors:* A list of contributors to the algorithm.
- *Description:* A high-level textual description of the algorithm functionality.
- *Algorithm Class:* The type of algorithm, based on a controlled taxonomy of algorithms.
- *Evaluation Metrics:* Possible evaluation metrics for validating input instances.
- *Source Code:* Links to code repositories of the algorithm.
- *Implementation languages:* A complete list of the languages, in which the algorithm is available.
- *Service Endpoint:* This URI depicts the location where the Linked API is executable.
- *Example Requests:* A list of URIs pointing to exemplary requests of the Linked API in any RDF serialization.
- *Example Responses:* A list of URIs pointing to exemplary responses of the Linked API in any RDF serialization.

Functional requirements consist of concrete inputs and outputs of the algorithm and pre- and postconditions of the execution:

- *Inputs:* The inputs of the algorithm are either resources of type *file* or resources of type *parameter*. These resources must provide further information about their data type, about the concepts occurring in the input, about the physical format and if they are required for the execution.
- *Preconditions:* Every input must be part of a precondition to be able to specify additional constraints the algorithm has on the inputs and additional features the input should have for the algorithm to work well.
- *Outputs:* The description of the outputs must have the same features as the one of inputs.
- *Postconditions:* The description of the postconditions must have the same features as the one of the preconditions. Features are to depict the implications on the output if the algorithm worked well.

In particular, the combination of pre- and post conditions, and algorithm class enable us to select adequate algorithms for completing a particular task (or task sequences). Central for our learning endeavours is stipulating evaluation metrics for an algorithm. This feature depicts if and how the system can automatically quantify results based on training samples or approximate them by certain variables.

Medical Linked APIs The basic idea is to make the interpretation algorithms (e.g. image classifiers, simulation algorithms or sensor interpretation techniques) available through APIs with ontological annotations. The algorithm itself can be hosted on any server and makes its functionality accessible via remote procedure calls. A so-called *Linked API* [19] then calls the server functions and extends the interface with a RDF description (i.e. ontological annotations) containing all the information presented in section 4.2. We call *Linked APIs* for medical assistance tasks *Medical Linked APIs* and will continuously refer to them as such in the rest of the paper. *Medical Linked APIs* provide their functional and non-functional requirements as part of this description.

We use the SMW as a repository of the available *Medical Linked APIs*. This is done by creating a page for each algorithm with all the information stipulated in section 4.2. The repository can either be browsed by people who want to test or reuse the *Medical Linked API* or by the execution engine.

The results computed by the *Medical Linked APIs* are directly saved in the semantic patient data store. Ontological annotations are automatically created comprising the inputs used, the outputs produced, the *Medical Linked API* employed and a time stamp.

4.3 Evidence-based Learning

The big picture of ensemble learning methodologies is being elaborated in [17]. The underlying assumption of combining algorithms for a specific task is that more algorithms might better generalize and generate more stable results. There are numerous different approaches to combine algorithms, e.g. boosting [5], bagging [1], stacked generalization [22] or mixture of experts [8].

We follow the vision of a self-adaptive learner using observed evidence, as presented in [21]. To optimize the results of a single task with given pre- and postconditions, we employ a technique inspired by both instance-based learning and ensemble learning to dynamically combine the results of available interpretation algorithms in a generic way.

The evidence the learner uses comprises annotated training samples. These might include pairs of surgical activities (e.g. <knife, cut, kidney>) and the matching phase (e.g. resection) or other quantifiable input output pairs. Based on the ontological annotation of the latter, the learner searches for the closest k training instances. Each possible interpretation algorithm has then to be executed on the k closest training instances. Since the true results for the training set is known, the learner can calculate weights based on the performance of the algorithms. The result for the current data instance is obtained by a weighted majority voting. Deriving the closest data instances is not trivial, since it is dependent on the used semantic annotation and the similarity metric. These decisions have to be ontologically specified for each scenario.

The learning component is also made available as ontologically as *Linked API* and is being referred to as *Evidence-based Learning Linked API*.

4.4 Execution Engine

The information needed to run a *Medical Linked API* is directly encoded in the ontological description. The precondition of a Linked API specifically states what data is needed to execute the algorithm and the algorithm class enables querying specific *Medical Linked APIs*. We leverage the declarative nature of the algorithm descriptions and execute the algorithms reactively on a data-driven basis. This implies that no workflows are manually defined and no explicit trigger is needed to run the algorithms.

We therefore use the Linked Data-Fu engine[20]. It enables defining rules to manage the interaction with resources on the Web (in our case the execution of *Linked APIs*) and to virtually integrate distributed data sources. As described earlier, the semantic patient data store is an XNAT instance with a RDF extension. We defined rules to browse through all ontologically annotated projects, patients and files to have all the patient-relevant data.

The preconditions of the *Medical Linked APIs* are used as if condition for the rules. The concrete instantiation of the precondition with the available ontologically annotated data is the input to the *Medical Linked API*, which is transmitted via a HTTP POST request. The Engine first gathers all data by applying the XNAT-specific rules and then executes all *Medical Linked APIs* whose preconditions are fulfilled by respective patients. If further data sources have to be integrated the engine has only to be enriched with more data-dependent rules. This also covers real-time scenarios, in which continuous data streams have to be polled.

Evidence-based Learning Linked API is integrated into the workflow via rules listening to feed back of *Medical Linked APIs*.

The Linked Data-Fu execution engine is made available as *Linked API* and will be referred to as *Linked Data-Fu Linked API*. We can dynamically stipulate the frequency how often the rules are to be checked and enable the manual triggering of workflows by end-users.

Figure 2 illustrates the interplay of all components and integrates both parallel configuration of phase recognition and sequential configuration of TPM image processing algorithms.

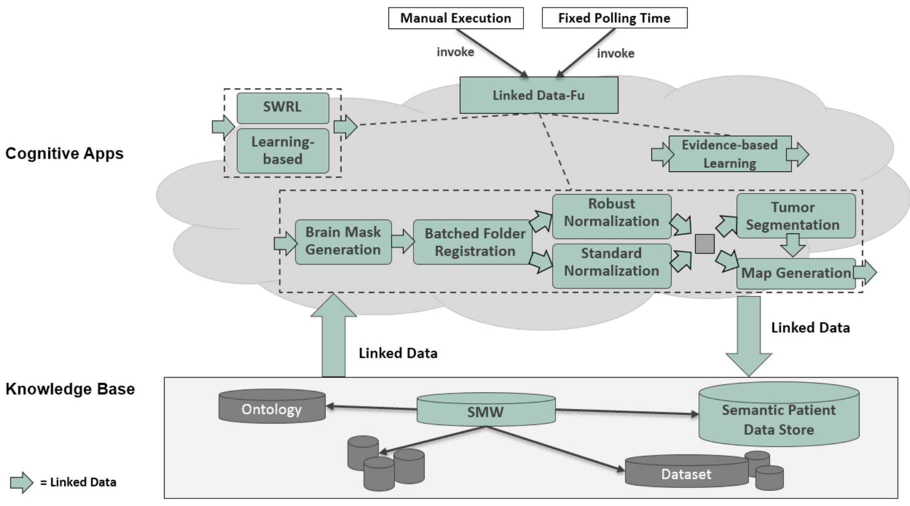

Fig. 2. Interplay of system components based on phase recognition and TPM image processing algorithms

5 Parallel Configuration of Surgical Phase Recognition Linked APIs with Evidence-based Learning

The scenario implementation comprises the execution and combination of two phase interpretation algorithms as introduced in the section 3.

5.1 Phase Recognition Linked APIs

We wrapped all our *Phase Recognition* algorithms as Linked APIs to make them permanently available and not rely on any particular platform or programming language. Therefore, algorithms are also directly addressable via a URI and the communication is handled via HTTP.

Here we provide two examples of *Phase Recognition Linked APIs* – two phase recognition algorithms wrapped to consume and produce RDF, which conveniently integrates the APIs with the knowledge base and the semantic patient data store. As we have to model pre- and postconditions based on the algorithm inputs and outputs, ontological annotations for the data was the prerequisite. Pleas see figures 3 and 4 for the pre- and postconditions for the *Learning-based Phase Recognition Linked API* in the RDF serialization TURTLE [4].

```
PREFIX sp:  <http://surgipedia.sfb125.de/wiki/Category/>
PREFIX lap: <http://surgipedia.sfb125.de/files/lapOnt.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

?trainingSample     rdf:type        sp:AnnotatedSurgery.

?ontology           rdf:type        sp:Ontology.

?event              rdf:type        sp:SurgicalEvent.
?event              sp:instrument   ?instrument.
?event              sp:action       ?action.
?event              sp:structure    ?structure.

?instrument         rdf:type        lap:Instrument.

?action             rdf:type        lap:InstrumentalProperty.

?structure          rdf:type        lap:TreatedStructure.
```

Fig. 3. Preconditions for *Learning-based Phase Recognition Linked API*

```
PREFIX sp:  <http://surgipedia.sfb125.de/wiki/Category/>
PREFIX lap: <http://surgipedia.sfb125.de/files/lapOnt.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

?phase              rdf:type        lap:Phase.
```

Fig. 4. Postconditions for *Learning-based Phase Recognition Linked API*

We observe the algorithm class *Phase Recognition* as part of the central algorithm taxonomy. The precondition states further that the algorithm needs a

[4] http://www.w3.org/TR/turtle/

laparoscopic ontology (in OWL.) and training samples of type *AnnotatedSurgery* (i.e. activity triples as introduced earlier with correct phase prediction). The pre- and postcondition of the *SWRL Phase Phase Recognition Linked API* are the same with the exception that the algorithm does not need training samples as input. *Learning-based Phase Recognition Linked API* only accepts annotated activities where instrument, action and treated structure have instances of concepts available in the laparoscopic ontology.

The *Phase Recognition Linked APIs* are registered in the SMW by creating a page of type *Phase Recognition* (i.e. an instance of the concept Phase Recognition). The complete set of ontological annotations introduced in section 4.2 is then available for people interested in reusing or testing the phase interpretation algorithms and for the execution engine.

The results of the *Phase Recognition Linked APIs* are directly saved in the semantic patient data store with ontological annotations about the input activity, the predicted phase, a time stamp and the used *Phase Recognition Linked API*.

5.2 Evidence-based Learning for Phase Recognition Linked APIs

As mentioned earlier, we use a weighted majority voting mechanism based on evidence in the system, i.e. training samples or validated predictions.

The similarity between two activities <instrument, action, treated structure> is 1 if the instruments, actions and structures are the same and if the last 3 activities are the same as well. If fewer past activities are the same the similarity linearly decreases.

All nearest neighbours with a similarity higher than $0,5$ are used to compute the weights for *SWRL Phase Recognition Linked API* and *Learning-based Phase Recognition Linked API*. The final decision is made based on a weighted majority voting.

5.3 Execution of Phase Recognition Linked APIs

The ontological annotation of the *Phase Recognition Linked APIs* are first translated into rules applicable for the Linked Data-Fu engine. Based on the preconditions the engine knows how to initialize the algorithms with a specific ontology and training samples (the latter are only needed for *Learning-based Phase Recognition*).

We than formulate additional rules stating that if ontologically annotated activity events are present then the initialized *Phase Recognition Linked APIs* must be executed. The *Linked Data-Fu Linked API* then periodically executes the rule on the prevalent activities. Finally, the *Evidence-based Learning Linked API* is being integrated into the worfklow based on the results of both *Learning-based Phase Recognition Linked API* and *SWRL Phase Recognition Linked API*, which are available in the semantic patient data store.

Although we tested the scenario with training samples stored in the semantic patient data store, this set up directly enables the intraoperative use case, in which the system is fed sensor data in real-time and directly predicts the current surgical phase.

5.4 Evaluation of Phase Recognition Scenario

The evaluation of the *Phase Recognition Linked APIs* and the parallel configuration through both *Linked Data-Fu Linked API* and *Evidence-based Learning Linked API* was conducted in terms of a performance benchmark, correctness and improvement of the results. We therefore compared the recognized phases of pancreas resections and measured the time the executions of the *Phase Recognition Linked APIs* needed, if the predicted phases are equal to local execution and what impact the *Evidence-based Learning Linked API* has.

The correctness and time performance of the *Phase Recognition Linked APIs* will show that – as claimed in the introduction – our concepts are eligible for medical interpretation algorithms. Both *SWRL Phase Recognition Linked API* and *Learning-based Phase Recognition Linked API* were executed automatically and in parallel with fulfilled preconditions which shows the automatic execution by using the *Linked Data-Fu Linked API* (claims (i), (ii)). The *Evidence-based Linked API* scores the results of both *Phase Recognition Linked APIs* by the available evidence (i.e. annotated surgeries) of the system (claim (iii)).

We cross-validated 5 annotated Pancreas surgeries by using 4 annotated surgeries as training samples and 1 annotation as target to predict the phases. As the *Learning-based Phase Recognition* algorithm is non-deterministic we had to average multiple runs.

All Linked APIs (i.e. *SWRL Phase Recognition Linked API*, *Learning-based Phase Recognition Linked API*, *Linked Data-Fu Linked API* and *Evidence-based Learning Linked API*) were implemented and then hosted on an university intranet.

Correctness The phase recognition algorithms have been developed for local use thus needed to validate if the results of the remote execution are correct. The cross validation was conducted on 5 annotated pancreas surgeries, PR_i for $i = 1, \dots 4$. The subsequent tables show the rate of true predictions for local execution and remote execution. We average the results of 5 iterations for each surgery PR_i to account for the non-deterministic behaviour of *Learning-based Phase Recognition Linked API.* We round to 4 decimal places.

Algorithm	$PR1$	$PR2$	$PR3$	$PR4$	$PR5$
Learning-based	$0,9062$	$0,6635$	$0,9032$	$0,4484$	$0,6383$
SWRL	$0,9315$	$0,7753$	$0,89$	$0,8137$	$0,7241$

Table 1. Local execution of phase recognition algorithms

Algorithm	$PR1$	$PR2$	$PR3$	$PR4$	$PR5$
Learning-based	0.9056	0.6596	$0,9$	$0,4443$	$0,6351$
SWRL	$0,9315$	$0,7753$	$0,89$	$0,8137$	$0,7241$

Table 2. Remote execution of *Phase Recognition Linked APIs*

The results for *SWRL Phase Recognition Linked API* are exactly the same as for local execution and the results for *Learning-based Phase Recognition Linked API* deviate minimally.

Performance benchmark We evaluated the usefulness of the semantic pipeline in terms of time performance. The system architecture is distributed and numerous HTTP requests have to be made. We calculated the overhead we are producing when sending data over the internet to show that our system can be applied to the intra-operative use case. The following values represent the time the HTTP POST and GET requests in milliseconds. We, again, averaged the values over 5 iterations of running through all pancreas surgeries. We rounded to 1 decimal place.

First, *Linked Data-Fu Linked API* downloads all necessary data. For *SWRL Phase Recognition Linked API* this is only the ontology (99,2 kb) while for *Learning-based Phase Recognition Linked API* the training samples have to be downloaded additionally (always 4 samples with an average aggregated size of 1,5 mb). *Ontology*, *samples* and *training* are the individual initialization steps for *Learning-based Phase Recognition Linked API. Samples* depicts the aggregated time for 4 samples to be loaded and *initialization* initializes the *Phase Recognition Linked APIs. SWRL Phase Recognition Linked API* does not provide a training mechanism. *Event* measures the time for sending a triple with instrument, action and structure information, and waiting for the prediction. The *reset* operation restarts the *Phase Recognition Linked APIs.*

Algorithm	Download	Ontology	Training	Samples	Initialization	Event	Reset
Learning-based	$4083,0$	$2843,0$		$9924,0$	$2349,0$	$39,7$	$36,7$
SWRL	$2030,0$	$2767,0$		0	$3242,0$	$20,3$	$26,0$

Table 3. Time overhead of *Phase Recognition Linked APIs*

The download step is the main overhead compared to local execution. Getting the files takes a few seconds in both cases because *Linked Data-Fu Linked API* first iterates through all files of the project in the semantic patient data store. In a real-time scenario a data source with only necessary contents can be integrated if needed.

The initialization for the local version of the learning-based phase recognizer takes $11516,3ms$ and for SWRL $4018,0ms$ which is a bit faster. Considerable overhead is only produced for the initialization step which can be made before the intraoperative scenario is starting. The residual time performances suffice for a real-time scenario as sending an event takes only $30ms$ on average.

Improvements of Learning The results of the *Evidence-based Learning Linked API* for ontologically annotated pancreaas surgeries are depicted below compared against both *SWRL Phase Recognition Linked API* and *Learning-based Phase Recognition Linked API*.

Algorithm	PR1	PR2	PR3	PR4	PR5
Learning-based	0.9056	0.6596	0, 9	0, 4443	0, 6351
SWRL	0, 9315	0, 7753	0, 89	0, 8137	0, 7241
Evidence-based	0, 9332	0, 7786	0, 9180	0, 7782	0, 7238

Table 4. Results of *Evidence-based Learning Linked API* compared to the *Phase Recognition Linked APIs*

The learner generates stable results very close to the optimal *Phase Recognition Linked API* and sometimes even better.

6 Sequential Execution of Data-driven Image Processing for TPM Linked APIs

The scenario for processing images consists of a pipeline of seven algorithms visualized in figure 1, with data-driven decision concerning the normalization step.

6.1 TPM Image Processing Linked APIs

Each of the algorithms has been provided with a ontological annotations as explained in [6]. Figure 5 shows the pre- and postconditions for the *Brain Mask Generation Linked API*, which takes as input a headscan and two reference images (a brain atlas mask and a brain atlas image). This processing step outputs the brain image and brain mask of the headscan.

6.2 Execution of TPM Linked APIs

We applied the Linked Data-Fu engine to the TPM use case in [16] showing that data-driven procedures decision can be declaratively implemented in combination with *TPM Image Processing Linked APIs*.

```
                PREFIX rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                PREFIX dc:         <http://purl.org/dc/elements/1.1/>
                PREFIX sp:         <http://surgipedia.sfb 25.de/wiki/Special:URIResolver/>
?inputImage       rdf:type    sp:Category-3AHeadscan.         | ?brainImage  rdf:type    sp:Category-3ABrainImage.
?inputImage       dc:format   "image/nrrd".                   | ?brainImage  dc:format   "image/nrrd".

?brainAtlasImage  rdf:type    sp:Category-3ABrainAtlasImage.  | ?brainMask   rdf:type    sp:Category-3ABrainMask.
?brainAtlasImage  dc:format   "image/mha".                    | ?brainMask   dc:format   "image/nrrd".

?brainAtlasMask   rdf:type    sp:Category-3ABrainAtlasMask.   |
?brainAtlasMask   dc:format   "image/mha".                    |
```

Fig. 5. Pre- and postconditions for *Brain Mask Generation Linked API* [16]

As the *TPM Image Processing Linked APIs* have been registered in the SMW with all their ontological annotations, we could translate the preconditions into rules for the Linked Data-Fu engine. We, again, stipulated rules for gathering all projects, patients and files of the semantic patient data store. New rules based on the preconditions of the *TPM Image Processing Linked APIs* have then been created.

As depicted in the TPM scenario overview (see figure 1) headscan of different stages of the patient's treatment process are available in the semantic patient data store. They have been situated in central data taxonomy and ontologically annotated with type *Headscan*. An atlas image and and atlas mark of the brain are both available in the semantic patient data store, which enables the *Brain Mask Generation Linked API* to be called by the *Linked Data-Fu Linked API* for every annotated headscan. A brain image and a brain mask are being produced and fed back to the semantic patient data store with adequate ontological annotations. Based on the enriched ontologically annotated data for the patient new *Medical Linked APIs* might be executable. The preconditions for the *Batched Folder Registration* are fulfilled and all brain images are used to register all images together in one step. The workflow proceeds and is entirely executed on a declarative basis with no globally defined procedure.

The main difference to the phase recognition scenario is the lack of real-time requirements, since newly annotated data is automatically processed after a fixed polling time set for the *Linked Data-Fu Linked API*.

6.3 Evaluation of TPM Scenario

As depicted earlier, we did not assume a real-time scenario to be necessary. The sequential data-driven pipeline has thus been evaluated in terms of correctness of the respective *TPM Image Processing Linked APIs* in [6] and correct execution of data-driven execution of declarative workflows in [16]. This extends the capabilities of parallel configurations with the automatic composition and execution of complex sequential workflows where no procedures have to be hand-crafted (claim (i)).

The evaluations concerning the duration of execution conducted in [6] showed a dependency on the internet bandwidth. The impact of the delays compared to local execution do not considerably impact our scenario, as the requirements are not that time-bound. Depending on the polling time of the *Linked Data-Fu Linked API* (which was set to five minutes) the workflow is started within that time frame. The *TPM Image Processing Linked APIs* then create the correct

TPM after 12 minutes on average and make the results available in the semantic patient data store.

7 Discussions and Lessons Learned

The here introduced configurable system is able to deal with both sequential and parallel scenarios. The ontological modelling of the interpretation algorithms and their data is shown to be sufficient for these configurations to be created. However, with more *medical Linked APIs* being developed and added to the configuration system, and limited central capabilities of adapting all ontological annotations among each other, not all sequential and parallel possibilities might be leveraged. The learning component (or *Evidence-based Learning Linked API*) is being executed based on multiple results created for a task but does prerequisite the same ontological annotations to be used. In addition, complex workflows (such as computing a TPM with multiple *TPM Image Processing Linked APIs*) are also only constructed if pre- and postconditions are exact matches. If another image processing pipelines with different Linked APIs is added to the system, it will be automatically executed but this maybe not be in combination with the existing *TPM Image Processing Linked APIs* because the ontological annotations might not match perfectly. This can be easily solved by adding a reasoning component to the system, which creates matches between different ontological annotations.

The learning component currently gradually generates more stable results with more *Medical Linked APIs*. Please note that ensembles or interpretation algorithms (e.g. classifiers) generally do not assume to provide better results than the single optimal algorithm. As there are numerous approaches, it might be interesting to use multiple learning techniques and choose the optimal combination there. In addition, training the *Medical Linked APIs* adaptively (similarly to boosting approaches) might optimize the results even more.

8 Conclusion

We propose a configuration system able to automatically create sequential and parallel workflows based on ontological annotations for algorithms. The system enables automatic execution of new algorithms, if they are made available as *Medical Linked APIs* and are registered in a central semantic repository (we use a SMW) through a rule-based engine. Learning components can be added to the system to optimize the results, when numerous *Medical Linked APIs* are available for the same task. We applied our approach to both *Phase Recognition* and *TPM Image Processing* algorithms to show the system capabilities. Our evaluation results show that the *Medical Linked APIs* are providing the correct results as they did for local execution and run in a reasonable amount of time. The *Linked Data-Fu Linked API* automatically executes both *Phase Recognition Linked APIs* and *TPM Image Processing Linked APIs* when necessary data sources are available and ontologically annotated (claims (i), (ii)). The

Evidence-based Learning Linked API provides stable results when one algorithm performs very poorly and beats choosing the optimal algorithm if both evidence and similarity metric work well (claim (iii)).

Acknowledgements. This work was carried out with the support of the German Research Foundation (DFG) within projects I01, A01, R01, S01 and I04, SFB/TRR 125 "Cognition-Guided Surgery".

Conflict of Interest Statement: Patrick Philipp, Darko Katic, Maria Maleshkova, Stefanie Speidel, Achim Rettinger, Benedikt Kämpgen, Rudi Studer, Rüdiger Dillmann, Anna-Laura Wekerle, Sascha Zelzer, Hannes Kenngott and Beat Müller declare that they have no conflict of interest.

References

1. Breiman, L.: Bagging predictors. Machine learning 24(2), 123–140 (1996)
2. Butcher, L.: Case study: How health it is changing the practice of oncologyusing data in new ways. Oncology Times 34(24), 29–30 (2012)
3. Clearly, K., Chung, H.Y., Mun, S.K.: Or 2020: The operating room of the future. Laparoendoscopic and Advanced Surgical Techniques (2005)
4. Ferrucci, D.: Build watson: an overview of deepqa for the jeopardy! challenge. In: Proceedings of the 19th international conference on Parallel architectures and compilation techniques. pp. 1–2. ACM (2010)
5. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: Computational learning theory. pp. 23–37. Springer (1995)
6. Gemmeke, P., Maleshkova, M., Philipp, P., Götz, M., Weber, C., Kämpgen, B., Zelzer, S., Maier-Hein, K., Rettinger, A.: Using linked data and web apis for automating the pre-processing of medical images. COLD (ISWC) (2014)
7. Gil, Y., Gonzlez-Calero, P.A., Kim, J., Moody, J., Ratnakar, V.: A semantic framework for automatic generation of computational workflows using distributed data and component catalogues. Journal of Experimental & Theoretical Artificial Intelligence 23(4), 389–467 (2011), `http://www.tandfonline.com/doi/abs/10.1080/0952813X.2010.490962`
8. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation 3(1), 79–87 (1991)
9. Joyce, J.P., Lapinsky, G.W.: A history and overview of the safety parameter display system concept. Nuclear Science, IEEE Transactions on 30(1), 744–749 (Feb 1983)
10. Katic, D., Wekerle, A.L., Grtner, F., Kenngott, H.G., Mller-Stich, B.P., Dillmann, R., Speidel, S.: Knowledge-driven formalization of laparoscopic surgeries for rule-based intraoperative context-aware assistance. In: Proc. of Information Processing in Computer Assisted Interventions (IPCAI). pp. 0–0 (2014)
11. Kranzfelder, M., Staub, C., Fiolka, A., Schneider, A., Gillen, S., Wilhelm, D., Friess, H., Knoll, A., Feussner, H.: Toward increased autonomy in the surgical or: needs, requests, and expectations. Surgical Endoscopy 27(5), 1681–1688 (2013), `http://dx.doi.org/10.1007/s00464-012-2656-y`

12. Lalys, F., Bouget, D., Riffaud, L., Jannin, P.: Automatic knowledge-based recognition of low-level tasks in ophthalmological procedures. International Journal of Computer Assisted Radiology and Surgery 8(1), 39–49 (2013), `http://dx.doi.org/10.1007/s11548-012-0685-6`
13. Neumuth, T., Jannin, P., Strauss, G., Meixensberger, J., Burgert, O.: Validation of knowledge acquisition for surgical process models. Journal of the American Medical Informatics Association 16(1), 72 – 80 (2009), `http://www.sciencedirect.com/science/article/pii/S106750270800193X`
14. Neumuth, T., Strau, G., Meixensberger, J., Lemke, H., Burgert, O.: Acquisition of process descriptions from surgical interventions. In: Bressan, S., Kng, J., Wagner, R. (eds.) Database and Expert Systems Applications, Lecture Notes in Computer Science, vol. 4080, pp. 602–611. Springer Berlin Heidelberg (2006), `http://dx.doi.org/10.1007/11827405_59`
15. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: Lessons in creating a workflow environment for the life sciences: Research articles. Concurr. Comput. : Pract. Exper. 18(10), 1067–1100 (Aug 2006), `http://dx.doi.org/10.1002/cpe.v18:10`
16. Philipp, P., Maleshkova, M., Götz, M., Weber, C., Kämpgen, B., Zelzer, S., Maier-Hein, K., Rettinger, A.: Automatisierte verarbeitung von bildverarbeitungsalgorithmen mit semantischen technologien. In: Bildverarbeitung fr die Medizin (BVM). pp. 0–0 (2015)
17. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33(1-2), 1–39 (2010)
18. Speidel, S., Benzko, J., Krappe, S., Sudra, G., Azad, P., Mller-Stich, B.P., Gutt, C., Dillmann, R.: Automatic classification of minimally invasive instruments based on endoscopic image sequences. In: Proc. SPIE. vol. 7261, pp. 72610A–72610A–8 (2009), `http://dx.doi.org/10.1117/12.811112`
19. Speiser, S., Harth, A.: Integrating linked data and services with linked data services. In: The Semantic Web: Research and Applications, pp. 170–184. Springer (2011)
20. Stadtmüller, S., Speiser, S., Harth, A., Studer, R.: Data-fu: A language and an interpreter for interaction with read/write linked data. In: Proceedings of the 22nd international conference on World Wide Web. pp. 1225–1236. International World Wide Web Conferences Steering Committee (2013)
21. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review 18(2), 77–95 (2002)
22. Wolpert, D.H.: Stacked generalization. Neural networks 5(2), 241–259 (1992)
23. Wood, I., Vandervalk, B., McCarthy, L., Wilkinson, M.: Owl-dl domain-models as abstract workflows. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies, Lecture Notes in Computer Science, vol. 7610, pp. 56–66. Springer Berlin Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-34032-1_6`