Journal of Web Engineering, Vol. 16, No. 5&6 (2017) 471–504

A SEMANTIC FRAMEWORK FOR SEQUENTIAL DECISION MAKING FOR JOURNAL OF WEB ENGINEERING

PATRICK PHILIPP, MARIA MALESHKOVA, ACHIM RETTINGER
Institute AIFB, Karlsruhe Institute of Technology
Karlsruhe, Germany
{patrick.philipp, maria.maleshkova, rettinger}@kit.edu

DARKO KATIC
HIS, Karlsruhe Institute of Technology
Karlsruhe, Germany
darko.katic@kit.edu

Current developments in the medical domain, not unlike many other sectors, are marked by the growing digitalization of data, including patient records, study results, clinical guidelines or imagery. This trend creates the opportunity for the development of innovative decision support systems to assist physicians in making a diagnosis or preparing a treatment plan. Similar conditions hold for the Web, where massive amounts of raw text are to be processed and interpreted automatically, e.g. to eventually add new information to a knowledge base. To this end, complex tasks need to be solved, requiring one or more interpretation algorithms (e.g. image- or natural language processors) to be chosen and executed based on heterogeneous data. We, therefore, propose the first approach to a semantic framework for sequential decision making and develop the foundations of a Linked agent who executes interpretation algorithms available as Linked APIs [43] on a data-driven, declarative basis [45] by integrating structured knowledge formalized with the Resource Description Framework (RDF), and having access to meta components for planning and learning from experience. We evaluate our framework based on automatically processing brain images, the ad-hoc combination of surgical phase recognition algorithms and experiential learning to optimally pipeline entity linking approaches.

Keywords: Sequential Decision Making, Linked APIs, Meta Learning, Medical Assistance, Entity Linking

1. Introduction

Growing degrees of digitalization in the medical domain drive the need for learning systems to integrate and interpret vasts amount of generated data. Physicians, for example, would benefit greatly from automatic procedures to interpret patient data, as their time is limited and the information space they have to deal with is diverse. In a similar vein, one evolving goal for the Web is to structure massive amounts of raw data, such as text, to be eventually able to construct complex queries against knowledge bases. As a consequence, automatic procedures are being developed to structure data without human intervention.

This trend opens up new challenges. Consider an image processing scenario which consists of several subtasks, having the goal of segmenting images to support end users (such as physicians) in their work. An image, first, has to be filtered for distorting elements and normalized in terms of color. The image, then, has to be aligned with other images in the knowledge base to ease interpretation. Depending on the request of an end user, one can

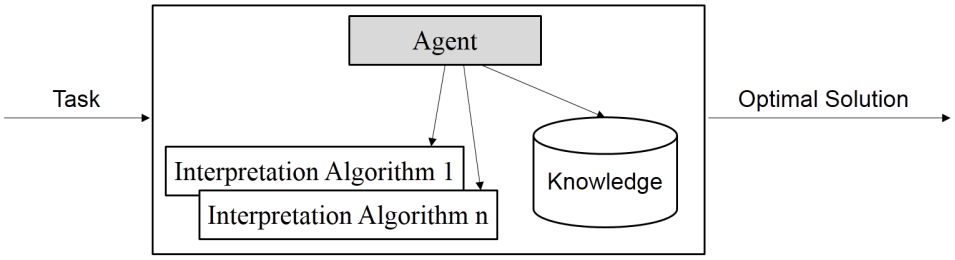

Fig. 1. An agent able to find optimal solutions for tasks.

eventually segment the image by using machine learning approaches.

These subtasks are very different in nature and require different steps to solve them. We refer to algorithmic approaches which solve such steps as interpretation algorithms, as the latter need to be able to interpret possibly complex state spaces to propose a solution and will assume access to a pool of such. There is no trivial way to automate processes for such complex tasks, as both end user requests and potential scenarios are diverse. With no domain expert available to support using an interpretation algorithm, it might be difficult to completely understand both parameters and functionality. Even more so, without substantial experience with the target domain, one might not be able assess the quality of such an execution. If, additionally, more than one interpretation algorithm might be eligible for solving a single subtask and produced outputs significantly vary in terms of quality, optimizing tasks quickly becomes intractable.

Such settings comprise highly heterogeneous tasks, interpretation algorithms and data:

- A slight change in a task might prerequisite completely different algorithms for a data point,
- Algorithms are based on different conceptual approaches and minimal parameter changes might have detrimental effect on their performances,
- Data points often come from different data distributions which makes processing and interpreting hard.

We argue that these challenges can be best dealt with by advances in *sequential decision-making* [39], where a centralized agent has to learn which actions to take in which world states. In our case, actions are interpretation algorithms, which have to be selected and executed for data points. This enables to re-use advances for prominent decision-theoretic frameworks such as Markov Decision Processes [35], which enable to model fine-grained world states, possible actions and uncertain outcomes. Our longterm goal is to develop a system (or agent) which knows how to optimally choose sequences of available interpretation algorithms for any given complex task, as illustrated in fig. 1.

We build on prior work in the Semantic Web which centers around semantically enriched Web services (so-called Linked APIs) [43, 44, 17] and their data-driven, declarative execution (with Linked Data-Fu [45]). Equipped with these powerful technologies, we now concentrate on building a framework to enable decision-making under uncertainty. Based on access to

interpretation algorithms for a complex task, one can develop *meta components* and easily plug them into the current workflow. These meta *components* can comprise any strategy to choose interpretation algorithms for a task.

Although these strategies do not have to be sophisticated, we especially focus on enabling adequately complex methodologies to be flexibly and easily used. We therefore base our work on problem settings in the medical- and Web domain where data sources, such as patient-related information or text available through news articles or social media, are heterogeneous as they often consist of different data distributions. We introduce different *meta components* for these domains which make use of available structured knowledge and training data (i.e. labelled past executions of interpretation algorithms) to select, weight and execute interpretation algorithms. Our proposed meta components are evaluated in terms of correctness and overall performance of selected interpretation algorithms, where – for the latter – we rely on labelled data sets.

Our contributions are threefold; we

(i) we propose and formalize a first approach to a semantic framework for sequential decision making, and enable flexible integration and testing of interpretation algorithms and meta strategies,

(ii) present two meta components for (sequential-) decision making in medical scenarios, namely a meta learner and an abstract planner, and

(iii) disclose novel challenges for sequential decision making for Natural Language Processing (NLP) in Web domains by integrating named entity recognizers and -disambiguators into our framework, therefore adapting prior meta components, developing a novel planner and composing all meta components for planning-related learning.

The remainder of this paper is structured as follows. We formalize the problem in section and show where our work has commonalities and differences to other approaches. The single components of our framework are, subsequently, being introduced in section . Section , then, integrates the components and shows how we solve complex tasks. In section , we show how our framework works in practise based on two medical scenarios and thereby dwell on a *meta learning* and an *abstract planning* component. Section deals with a Web scenario and presents novel meta components for automatically solving it. We discuss current developments and possible improvements in section and conclude the paper with a summarizing remarks (Sec.) and future work (Sec.).

2. Problem Formulation & Related Work

We, first, formalize the essentials of the problem of solving complex tasks in heterogeneous environments and disclose our core challenges. We then summarize the work related to ours and point out our contributions.

2.1. *Problem: Solving Complex Tasks with Access to Heterogeneous Interpretation Algorithms*

Let X be the set of all tasks, Y the set of all abstract tasks and A the set of all available interpretation algorithms. Let x^k be the k^{th} subtask of task $x \in X$. Let further S be the set

of *abstract states* defined by a subset of objects O, literals L and relations R. We denote, for simplicity, F_{s_k} as the set of features of a state s_k (i.e. a subset of $O \times R \times O$ and $O \times R \times L$). A grounded state $g(s_k)$ depicts an instance of s_k in nature. The set A_{s_k} defines the subset of applicable interpretation algorithms in s_k which is known to some degree. We, thus, assume that an interpretation algorithm $a_i \in A$ can be defined by a subset of features of F in a similar way as states $s_k \in S$. Knowing A_{s_k} depends on how we define features $f \in F$ for s_k and a_i. Let $T(s, a, s')$ be the transition function for some state s and interpretation algorithm a ending in s'. Our knowledge of $T(s, a, s')$, again, depends on the available features for s, a and s'. $T(g(s), a, g(s'))$ is not known and requires further knowledge to be approximated. A task $x(g(s_0), s_K)$ is a function defined on a grounded start state $g(s_0)$ and an abstract goal state s_K. Reaching an unknown grounded goal state $g(s_K)$ takes 1 to n state transitions $(g(s), a, g(s'))$. To solve $x(g(s_0), s_K)$, we need to find a sequence of interpretation algorithms a_i ending in the unknown grounded goal state $g(s_K)$ with high probability. An abstract task $y(g(s_0), s_K)$ is defined similarly but we need to find any sequence $a_1, \ldots, a_n$ to get from $g(s_0)$ to s_K. Our setting is much related to a Markov Decision Process (MDP) [35] (S, A, T, R, γ) with $R(s, a)$, in addition, being the reward function for state, interpretation algorithm pairs (s, a) and γ the discount factor. The latter regulates the influence of future interpretation algorithms a_i taken in future steps s_k on the value estimations of current states and actions. R quantifies the correctness of executing a in s and the value of the resulting state s' (based on $T(s, a, s')$) is defined by $V(s)$, the value function for states. Note that estimating $R(g(s), a)$ for $x(g(s_0), s_K)$ is not straightforward as $g(s_K)$ is unknown. To specify the goal s_K of a task, an absorbing state with $R(s_k, a_i) = 0$ has to be artificially modelled.

We define **abstract planning** as trying to solve an *abstract task* $y(g(s_0), s_K)$. Here, we ignore that multiple interpretation algorithms a_i might be available for s_k and only use conceptual information about a_i and s_k. **Meta learning** deals with the case of $|A_{g(s_k)}| > 1$ (i.e. more than one eligible interpretation algorithm) and tries to solve a subtask $x^k_{A_{g(s_k)}}(g(s_k), s_k + 1))$ to find the optimal output (i.e. the combination of multiple a_i) for $g(s_k)$. **Planning** deals with solving $x(g(s_0), s_K)$ with known T and R, and **planning-related learning** considers T, R unknown and tries to approximate them (as, for instance, is done in model-based reinforcement learning [46]).

We develop and apply abstract planning and meta learning techniques for our medical applications (see section), and will present a planning-related learning approach for a Web scenario in section . Based on the problem setting, we derived the following needs for our semantic sequential decision making framework:

Need for a Controlled & Semantic Vocabulary. If we want to have tasks automatically executed based on state-goal pairs; tasks, data and algorithms need to be using a common vocabulary. In addition to a shared language, semantic annotations enable to adequately document results of executions (i.e. generate provenance information) which might eventually enable reusing and integrating results from other tasks and institutions.

Need for Accessibility & Scalability. The pool of algorithms needs to be accessible in real-time and available for many concurrent tasks. New interpretation algorithms should be readily available to be used and evaluated.

Need for Data-driven & Declarative Execution. With a large number of available in-

terpretation algorithms for a state s_k, it will quickly become intractable to manually define and evaluate all possible permutations. By executing interpretation algorithms when they match s_k, we gain flexibility and can delegate the optimization problem. With growing experience, one could generalize the learned optimal decisions to similar s_k.

Need for Meta Learning Components. As the optimization problem is neither trivial nor homogeneous, it might not be solvable by a single piece of software. One rather needs several meta components which are experts for different s_k.

2.2. *Related Work*

We divide related works into (i) description languages for web services, (ii) frameworks for (sequential-) decision-making, (iii) workflow systems and (iv) machine-learning methods for workflow pipelines.

2.2.1. *Semantic Web Service Description Languages*

While there is a considerable body of works covering non-semantic Web service descriptions (eg. WSDL [7]), we restrict the comparison to semantic descriptions, as the latter depict an important step towards machine-readability and are the closest to our work (find an overview in [51]).

SAWSDL [26] is a semantic extension to WSDL (based on WSDL-S [a], an early semantic extension for WSDL), which however neglects to describe functional relationships between inputs and outputs. To this end, WSMO [37] and OWL-S [27] are based on OWL, enabling to model richer descriptions. However, both prerequsite complex modelling even for simpler services and – similar to WSMO-Lite [53] which extends SAWSDL with conditions and effects – suffer from inexact input/output mappings as pointed out by [44]. We, thus, rely on Linked APIs [43, 44] – an approach to Semantic Web services – where pre- and postconditions are modelled as graph patterns, and rely on the Minimal Service Model (MSM) description language [25] [b], an ontology with base elements to align with Linked APIs.

Closely connected to service languages is the OPMW (Open Provenance Model for Workflows) [16] ontology, which is based on the Open Provenance Model (OPM) [c]and provides essential elements to describe workflow-related provenance metadata. As it might be used to document the outcomes of Linked API executions, it does not compete with our approach but rather potentially extends our provenance metadata.

2.2.2. *Decision-Making Frameworks & Applications*

The *Stanford Research Institute Problem Solver* (STRIPS) [13] comprises all functionalities for problem solving, i.e. a formal language to describe actions (i.e. algorithms) and a planner to choose actions given states. STRIPS defines actions which consist of an add-list, a delete-list and a precondition. By adding or deleting state conditions, one can express impacts of

[a] https://www.w3.org/Submission/WSDL-S/
[b] http://iserve.kmi.open.ac.uk/ns/msm/msm-2014-09-03.html
[c] http://open-biomed.sourceforge.net/opmv/ns.html

actions. STRIPS is the baseline for multiple decision-making methods, such as Problem-solving Methods or Markov Decision Processes, which we now integrate into our work.

Problem-Solving Methods (PSMs) [34, 4] are closely connected to STRIPS as well as to our work. PSMs describe highly parameterized algorithms which, in conjunction with expert knowledge, are able to solve real world tasks. As in our work, PSMs are described in terms of functional specifications (i.e. the interpretation algorithm class), requirements (i.e. the semantic descriptions modelled with the Resource Description Framework (RDF) to understand the interpretation algorithm) and operational specifications (i.e. the pre- and postconditions). To this end, the *Unified Problem-solving Method Language* (UPML) [11] enables to graphically model and semantically describe PSMs, which supports both reusing and adapting available PSMs. While several PSMs might easily be pipelined to solve tasks consisting of several subtasks, possible future impacts are not taken into account and decision-theoretic frameworks, such as Markov Decision Processes, are applicable. Moreover, combining exchangeable PSMs might enable the development of more robust approaches, which we study in the meta learning problem.

Usually, planning (see [19] for an overview) involves learning a policy which, for each possible world state, suggests the optimal action. For tasks involving uncertainties with respect to action outcomes, stochastic STRIPS enables to assign probabilities to actions. To this end, Markov Decision Processes (MDPs) [35] deal with learning the parameters of stochastic STRIPS tasks, which we use as decision-theoretic baseline for our framework.

Planning techniques have been applied to Semantic Web services before (see [24] for an extensive overview). Approaches to enable dynamic orchestration of Semantic Web services comprise – among numerous others – Hierarchical Task Networks (HTN) [41, 23] or OWL reasoning [40]. MDPs have been applied to compose non-semantic Web services [9], but the reward function was solely dependent on static service level agreements and not processing or interpretation outcomes as available in our case. There are, to the best of our knowledge, no works for Semantic Web services and MDPs for such reward structures, where labelled past interpretation algorithm executions are used as feedback for the system to learn.

2.2.3. *Workflow Systems*

There is an ongoing research in so-called 'workflow systems' that enable describing and executing algorithms of different kinds.

Pegasus [8] is a workflow management system able to map abstract pipelines for simulation data analysis to distributed computing environments. In a similar vein, FireWorks [21] focuses on enabling workflows to be executed on supercomputers. AKSALON [10] enables to define grid workflow applications via a graphical user interface or directly via XML, thereby easing the use of grids. dispel4py [14] is a Python-based framework to enable workflows for data streams, especially for distributed computing. Makeflow [1] enables to define workflows based on Unix Make, which are suitable for data-intensive distributed computing applications. Finally, web service workflows are approached by the LanguageGrid project [d], where diverse NLP interpretation algorithms as integrated from various institutions. All above workflow systems are sophisticated approaches to pipeline services, but neither focus on automatic decision-making for single data points nor exploit semantic annotations.

[d] `http://langrid.org/`

Taverna [30], on the other hand, is a scientific workflow system supporting process prototyping by creating generic service interfaces and thus easing the integration of new components. Lightweight semantic descriptions modelled in RDF are being used to better capture the view of the scientists. Taverna is also able to integrate data from distributed sources and automate the workflow creation process for users. Taverna was successfully used in the PANACEA project [e]for NLP tasks. Our work is strongly related to Taverna in terms of using RDF to describe (NLP-) services. Service selection then equals constructing SPARQL queries, where our framework diverges from Taverna with regards to using Machine Learning to optimize the selection for single data points. Similarly, Galaxy [6] enables to conduct sustainable and reproducible workflow experiments based on well-defined services and been successfully used for the LAPPS Grid project [f] where numerous NLP services have been implemented, semantically described using JSON-LD [g]and published on the Web. Our work shares many overlaps with both Taverna and Galaxy, but differs in the use of meta components to combine and plan under uncertainty. Our framework would greatly benefit from reusing semantically annotated NLP services of prior projects.

The work centered around semantic workflows [18] aims to enable the automatic composition of components in large-scale distributed environments. Generic semantic descriptions support combining algorithms and enable formalizing ensembles of learners. Therefore, conditions and constraints need to be specified. The framework also automatically matches components and data sources based on user requests and introduced prior mentioned OPMW ontology for provenance generation. We, also, employ artificial intelligence planning techniques but build our work on the decision-theoretic framework of Markov Decision Processes (MDPs), which are based on environmental feedback and enable approaching prior defined *planning-related learning* problem. We thus rely on labelled past executions, which might be annotated by OPMW.

Wood et al. [54] create abstract workflows as domain models which are formalized using the Web Ontology Language (OWL) and enable dynamic instantiation of real processes. These models can then be automatically converted into more specific workflows resulting in OWL individuals. The components can be reused in another context or process, and one can share abstract representations across the Web through OWL classes. The authors also rely on triple patterns to select appropriate annotated components for subtasks. As only subclass relationships are exploited, conceptual generalization might be achieved only through the ontology and dealing with uncertainties is not directly approached. We, on the other hand, target the joint problem of both enabling conceptual generalization as well as grounded generalization, where pipelines are optimized based on individual characteristic of data points.

Finally, automatic orchestration of analytical workflows is studied in [5]. The system essentially uses a planner, a leaner and a large (structured-) knowledge base to solve complex tasks. A large amount of potential workflows are taken into account to answer a user specified query with the optimal choice. The decision process comprises complex learning and planning approaches, and entails exploring large possible feature spaces. Lastly, atomic actions are

[e]http://www.panacea-lr.eu/
[f]http://www.lappsgrid.org/
[g]http://json-ld.org/

lifted with semantic annotations to better adapt to user queries. The proposed system, similar to our work, assesses interpretation algorithms in terms of labelled data and uses a planning engine to compose interpretation algorithms given a user query. The system, however, does not enable flexibility in meta components and thus presumes manually fixed meta learners and meta planners. In addition, weights for interpretation algorithms are not re-weighted for single data points, as the system relies on global ensemble functions.

We, finally, revisit work on pure machine learning approaches to optimize workflows using labelled past executions.

2.2.4. *Machine Learning-based Optimizers for Workflows*

Nguyen et al. [29] define the Machine Learning task of Meta-Mining, where optimal data mining workflows are constructed based on labelled data. The authors extend the problem of meta learning, where models learned for one-step tasks, such as classification or regression, are transferred to new, unseen data sets. The central difference to our work is to optimize with regards to complete data sets, not fine-grained single data points. Even more so, the learned workflows remain static for data sets, while there might be better solutions.

The problem of Automating Machine Learning (AutoML) [48] also deals with optimizing data mining workflows and was originally stated by [36]. The problem is, for example, approached auto-sklearn [12] and entails automating feature selection, interpretation algorithm selection and hyperparameter optimization. The authors leverage Bayesian optimization techniques but also make use of ensemble- and meta learning. Our work, however, focuses on selecting interpretation algorithms for single data points, where multiple instances of the same interpretation algorithm with different parameterizations might be used.

2.2.5. *Overall Summary of Related Works*

To sum up our comparison to all related works, we use a data-driven approach to execute workflows and work towards completely automatic, declarative and optimal compositions of such. Our novel contribution essentially enables to develop powerful agents capable of solving complex tasks in heterogeneous environments. Besides, only a small fraction of the above approaches employ a structured knowledge base. We are able to store structured performance-related information and try to reuse this evidence in order to optimize results. In addition, we flexibly integrate new kinds of structured knowledge as well and use so-called *meta components* to optimize decision making under uncertainty based on Linked Data-Fu [45].

3. Components of the Semantic Framework for Sequential Decision Making

Our system infrastructure comprises four core component types, where *Linked* denotes that the respective component exploits semantic annotations according to our framework.

1. A Structured Knowledge Base
2. Linked Interpretation Algorithms
3. Linked Meta Components

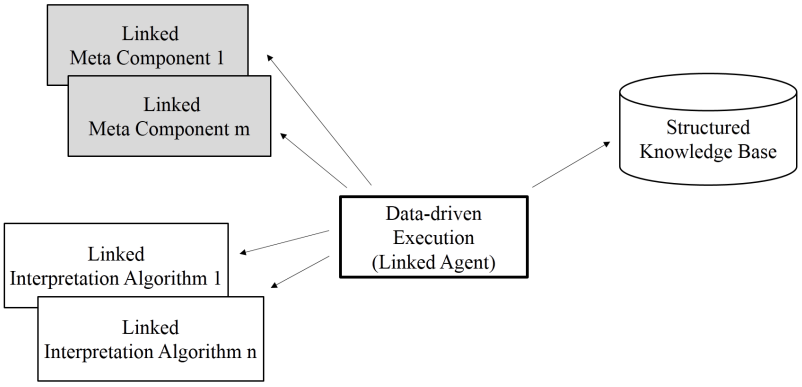

Fig. 2. Schematic Overview of the Framework.

Table 1. Minimal description for Linked interpretation algorithms.

Non-functional requirements	**Functional requirements**
Domain Experts	*Inputs*
Service Endpoint	*Preconditions*
Example request & response	*Outputs*
Algorithm class	*Postconditions*

4. A Data-Driven Execution Engine (Linked Agent)

Fig. 2 illustrates the framework components. We will now explain their respective functionality and integrate them in section .

3.1. *Structured Knowledge Base*

A structured knowledge base stores both metadata and data, and provides a common and controlled vocabulary modelled with RDF. As the Linked Data principles suggest, persistent URIs to resources have to be available and provide sufficient information for lookups. Appropriate concepts for interpretation algorithms and data were modelled to enable the integration of new components. Fig. depicts the components of the structured knowledge base used in our medical scenarios.

3.2. *Linked Interpretation Algorithms*

We deploy interpretation algorithms as Web services to make them easily accessible in our infrastructure and follow the idea of Linked APIs [43, 17]. A *Linked interpretation algorithm* (an interpretation algorithm lifted to a Linked API) provides a standardized description of its functionality by reusing elements of the structured knowledge base. The description also defines how to communicate with the Linked interpretation algorithm and how to execute its methods. A minimal set of information of the description is summarized in table 1.

An intuitive example of an arbitrary image processor is given in fig. 3. An image, defined in a data type ontology, is part of a pre- and postcondition of a Linked interpretation algorithm. Pre- and postconditions define strict rules about the states before- and after executing

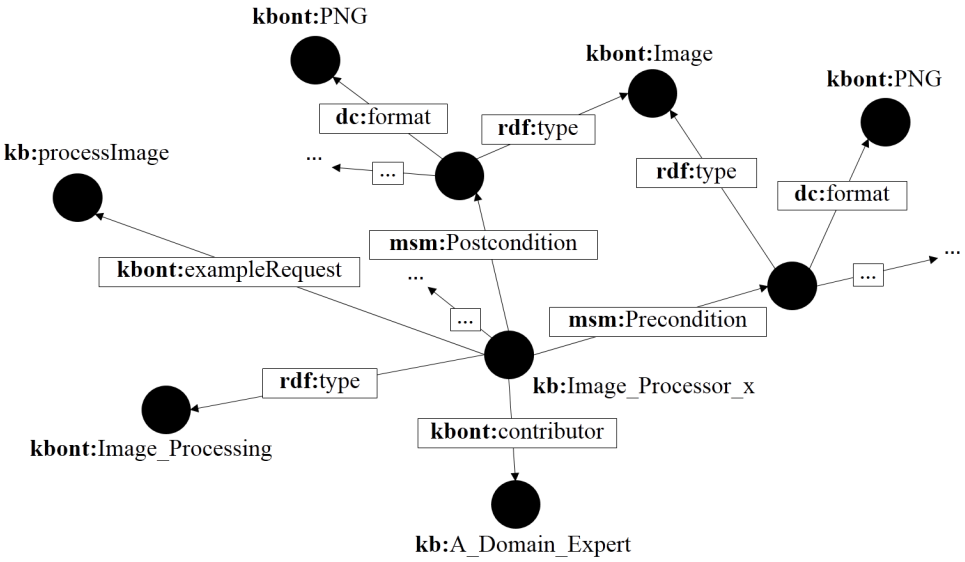

Fig. 3. An exemplary Linked image processor (namespaces omitted).

the Linked interpretation algorithm. The degree of detail of both pre- and postcondition is strongly dependent on the wrapping process of the respective interpretation algorithm. If semantics and interpretation algorithm are strongly intertwined, fine-grained semantics with rich information are available.

We use **kb** and **kbont** namespaces to describe instances and ontologies available in the knowledge base but will point out which exact ontologies we used in the respective scenarios. The **msm** namespace [h]corresponds to the minimal service model [25] which advocates and enables lightweight semantics for Web services.

3.3. *Linked Meta Components*

There might be numerous approaches to choose, on a meta level, among outputs of Linked interpretation algorithms given $g(s_k)$. Such strategies generate hypotheses which can be naive, sophisticated, biased on subjective criteria or otherwise. We enable flexible using, testing and exchanging of potentially powerful meta approaches in terms of so-called *Linked meta components*. Linked meta components are essentially Linked APIs and implement any decision making strategy of arbitrary complexity. To have access to all available interpretation algorithms, we assume a structured knowledge base linking to them.

A Linked meta component specifies the amount of information it needs by its precondition and is only called if the agent can provide for all information. Besides a list of all available interpretation algorithms, a learner might, for instance, want to access a performance table which stores a history of validated results. We will discuss four cases of meta components, namely *abstract planning*, *meta learning*, *planning* and *planning-related learning* in section and show their practical application in sections and .

[h]http://iserve.kmi.open.ac.uk/ns/msm/msm-2014-09-03.html

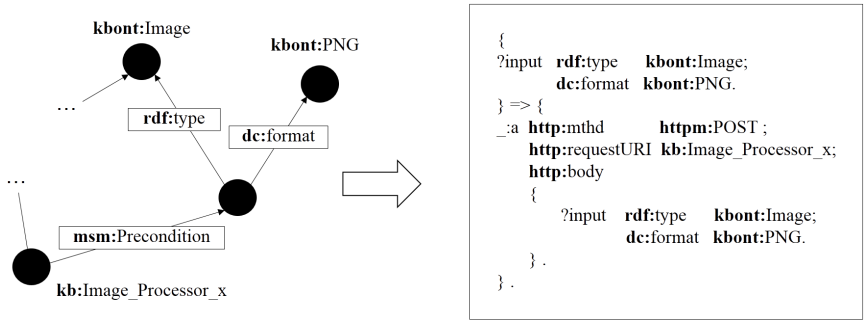

Fig. 4. A Linked Data-Fu rule for the Linked image converter.

3.4. *Data-driven Execution (Linked Agent)*

We integrate the prior components by using Linked Data-Fu [45]. The Linked Data-Fu rule-based execution engine describes and implements a formalism to virtually integrate data from different sources in real-time and have Linked APIs executed based on rules. In our framework, Linked Data-Fu searches eligible Linked meta components and Linked interpretation algorithms for (newly arrived) annotated data, and uses the structured knowledge base to execute them. Each Linked interpretation algorithm is represented as single rule which we automatically generate based on its description. After a Linked Interpretation Algorithms or Linked Meta Component has been executed, the grounded semantic annotations for its preconditions and postconditions are stored in the Structured Knowledge Base and serve as provenance information, which comprise additional timestamps. Fig. 4 summarizes this process. When a state $g(s_k)$ fulfills the preconditions of a Linked interpretation algorithm, a HTTP POST request with grounded preconditions is issued to its service URI. While this enables automatically executing Linked interpretation algorithms for numerous use cases, there are complex preconditions which prerequisite partial groundings, as will be discussed in section .

The automatic matching between Linked interpretation algorithms and structured data is highly advantageous. If interpretation algorithms have to be trained by samples, the agent can directly feed all annotated training data to the Linked interpretation algorithm. This is generally possible by merely defining rules for Linked Data-Fu. Even more important, with a growing number of diverse Linked interpretation algorithms, we can automatically solve new complex tasks without additional manual effort.

We refer to the Linked Data-Fu engine instantiated in our framework as *Linked agent*. As common in sequential decision-making in centralized systems, a single agent controls the decision process and chooses action (i.e. interpretation algorithms) given states. In our case, the agent exploits Linked meta components for interpretation as well as N3 rules for execution to take decisions.

Linked Interpretation Algorithms and Linked Meta Components can be directly discovered by their type as well as pre- and postconditions. However, a good selection of the latter is based on two aspects, namely finding a *conceptually eligible* component or interpretation algorithm and finding a *well performing* component or interpretation algorithm given the current data point. Both aspects might be approached by keeping and using appropriate

provenance metadata about their performances for different tasks, based on which agents have to learn and decide which interpretation algorithms and components to choose.

4. Solving Complex Tasks with Linked Meta Components

We will now explain the required interplay between the components of the semantic framework to enable sequential decision making for complex (abstract-) tasks $y(g(s_0), s_K)$ and $x(g(s_0), s_K)$. We, first, address the Linked abstract planning case where an abstract task $y(g(s_0), s_K)$ has to be solved by finding appropriate Linked interpretation algorithms a_i . Here, we do not try to distinguish between high- and low quality results (as this involves checking the instances), and trust the descriptions of Linked interpretation algorithms. Each Linked interpretation algorithm applicable in $g(s_k)$ *and* needed for reaching s_K will be chosen. The second case deals with Linked meta learning and builds on Linked interpretation algorithms selected by Linked abstract planning. An 'optimal' output of Linked interpretation algorithms is returned for $g(s_k)$ if $|A_{g(s_k)}| > 1$ (as otherwise there is no choice to make). In the planning setting, both the rewards R of Linked interpretation algorithms and possible transitions T are known (e.g. because of a Linked meta learners and abstract planners) but a grounded plan has to be generated. The final case deals with planning-related learning where R and T are unknown and prior meta components have to be reused and extended to deal with uncertainties.

In fig. 5 we give a generic overview of interactions between the Linked agent and Linked meta components, Linked interpretation algorithms and the structured knowledge base. The Linked agent first queries the knowledge base to get all available Linked interpretation algorithms and then calls a Linked abstract planner. It executes the resulting set of Linked interpretation algorithms for reaching goal s_K and subsequently calls a Linked meta learner capable of assessing and combining the generated candidate outputs. In section , we discuss the case where this workflow has to be repeated multiple times, as more than one subtask x^k exists.

4.1. *Linked Abstract Planning*

Depending on a new task $y(g(s_0), s_K)$, the Linked agent evaluates the grounded state $g(s_k)$ in terms of rule checking. The Linked agent, therefore, keeps a set of automatically generated rules for each Linked interpretation algorithm. We cannot assume, however, that $g(s_k)$ only triggers interpretation algorithms which help reaching the goal s_K, as there might be a large amount of Linked interpretation algorithms. We, thus, need a Linked abstract planner to only return candidate Linked interpretation algorithms for reaching s_K.

A Linked abstract planner has to first query the structured knowledge base for all available Linked interpretation algorithms. It, then, decides which Linked interpretation algorithms are applicable to reach s_K based on an arbitrary mechanism and finally outputs a subset of A. Fig. 5 illustrates the agent rule to call the Linked abstract planner depending on its precondition. We describe one implementation of a Linked abstract planner in section .

4.2. *Linked Meta Learning*

In the meta learning setting, we want to find an optimal output based on several Linked

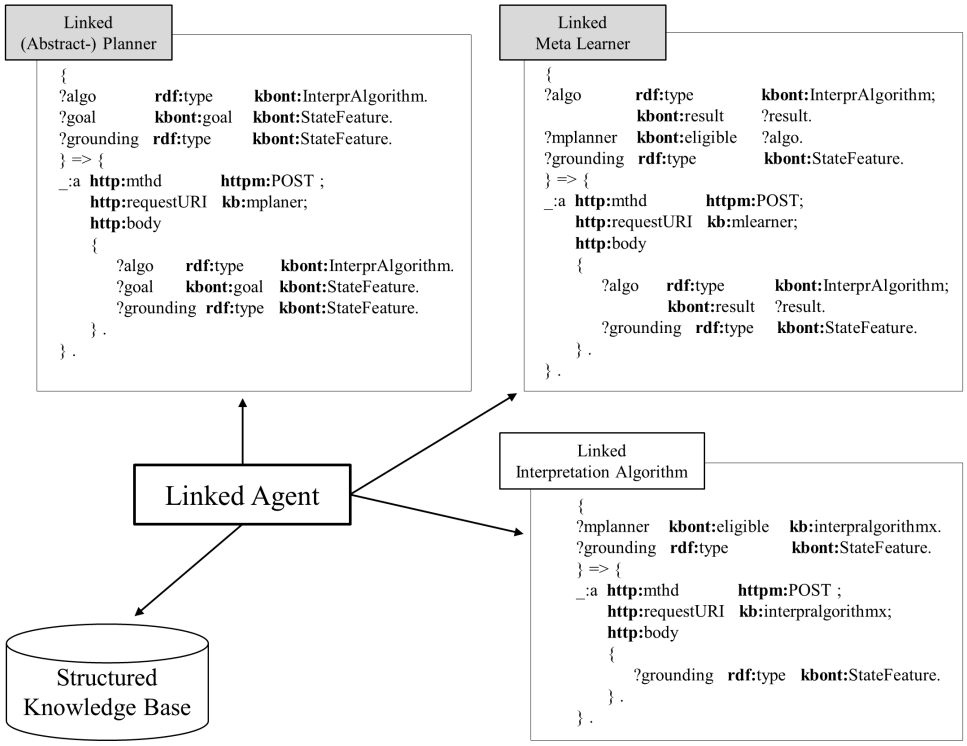

Fig. 5. Interactions within the framework to integrate Linked meta components.

interpretation algorithms a_i solving the same subtask $x^k(g(s_k), s_{k+1})$. Eligible Linked interpretation algorithms candidates $A_{g(s_k)}$ to reach s_{k+1} are known (e.g. due to a Linked abstract planner), but one is still faced with uncertainty about their performances given $g(s_k)$. Solving the learning setting can be approached with simple heuristics, but might require complex machine learning approaches which exploit past labelled executions of Linked interpretation algorithms available in the Structured Knowledge Base to give good estimates. With our framework, we enable using any meta learning approach by wrapping it as Linked meta learner.

Fig. 5 describes the generic rule to execute any available Linked meta learner. Section introduces a Linked meta learner for medical phase recognition which can be reused for other tasks, such as named entity recognition and -disambiguation. Multiple outputs of Linked interpretation algorithms are combined (e.g. by a weighted majority voting) after all eligible Linked interpretation algorithms have been executed. The linked meta learner, thus, expects results of Linked interpretation algorithms to compute weights.

Linked meta learners could also be used to select Linked interpretation algorithms before executing them based on their predicted performance. This might be crucial if a task has constraints, such as a budget in terms of number of executions. Reasons for execution budgets include the prevalence of cost models for available Linked interpretation algorithms or the massive use of resources (as is often the case for image processing algorithms).

4.3. *Linked Planning*

Other than in abstract planning, planning requires information about state- and interpretation algorithm groundings other than for the starting state $g(s_0)$. Linked planners are needed when pre- and postconditions of a Linked API do not suffice to solve a task, as state representation are too complex. This is the case for our named entity recognition and -disambiguation scenario presented in section .

A Linked planner is to be invoked before Linked interpretation algorithms are executed, i.e. under the same conditions as Linked abstract planners. However, we might need to replan several times for solving a grounded task $y(g(s_0), s_K)$, namely after several grounded subtasks g^k have been solved. To enable a data-driven execution of this process, we need to ground preconditions based on available data and Linked interpretation algorithms, and reason about uncertainties (as is done in Linked planning-related learning).

4.4. *Linked Planning-Related Learning*

As defined in section , planning-related learning entails estimating both R and T. Developing Linked planning-related learners entails using Linked meta learners, -abstract planners and planners, extended with uncertainty handling. One has to keep probability distributions about the states $g(s_k)$ (e.g. based on hypotheses by Linked meta learners), as one is uncertain about their correctness. We deal with Linked planning-related learning in section to optimize and automate the task of entity linking.

An extension of planning-related learning might be incorporating constraints for executing interpretation algorithms, such as cost or time budgets. Then, one has to deal with further challenges, such as dealing with more incompleteness in data (i.e. one does not know about the results of several interpretation algorithms).

5. Medical Scenarios with Meta Components & Evaluations

We now introduce two scenarios set in the medical domain and illustrate possible Linked meta components. The first scenario deals with image processing, where a number of image processing interpretation algorithms need to be pipelined to derive a so-called brain tumor progression mapping. We developed a Linked abstract planner to derive eligible Linked interpretation algorithms for the so-called brain tumor progression mapping (TPM). In the second scenario, we optimized the choice among two Linked interpretation algorithms for phase recognition in minimal invasive surgeries with a Linked meta learner. Both scenarios use a common instantiation of our framework. We will start by explaining the shared components and subsequently focus on the individual scenarios.

5.1. *A Semantic Framework for Medical Sequential Decision Making*

The medical framework with its interpretation algorithms and data is being developed within the Cognition-Guided Surgery project [i]. Every interpretation algorithm considered in the scenarios was wrapped as Linked interpretation algorithm. We modelled the descriptions with domain experts and developers of the interpretation algorithms, and integrated them in a central instance of a Semantic MediaWiki (SMW). To this end, all semantic annotations were based on MeSH (Medical Subject Headings) [j], RadLex [k], SNOWMED-CT [l] and FMA (Foundational Model of Anatomy) [m].

We use an instance of XNAT [n] to store patient-relevant data and provide a RDF wrapper which lifts XNAT with semantic concepts. The knowledge base can be considered as union between the SMW and its links to other resources, such as XNAT.

Linked interpretation algorithms can automatically be executed with the Linked agent. We implemented a conversion mechanism from Linked interpretation algorithm descriptions to Linked agent rules (see fig. 4), and, thus, reduced the manual work for integrating new Linked interpretation algorithms. The Linked agent crawls the hierarchy imposed by XNAT according to simple rules and executes every Linked interpretation algorithm per patient if it is eligible. While this only covers offline scenarios, we can easily extend the setting to the online case. The complete framework with two Linked meta components is illustrated in fig. 6.

5.2. *Tumor Progression Mapping in the Semantic Framework*

Tumor Progression Mapping (TPM) is an approach to visualize brain tumors in their progression over time. One, thereby, focuses on supporting radiologists in their daily work. Radiologists, otherwise, would have to assess the irregular growth of brain tumors based on raw headscans which causes a lot of extra effort. When generating a TPM, different types of images are used and produced, and adequate interpretation algorithms need to be executed in correct order and with correct subsets of images.

[i] http://www.cognitionguidedsurgery.de/
[j] https://meshb.nlm.nih.gov
[k] https://www.rsna.org/RadLex.aspx
[l] http://www.snomed.org/snomed-ct
[m] http://si.washington.edu/projects/fma
[n] http://www.xnat.org/

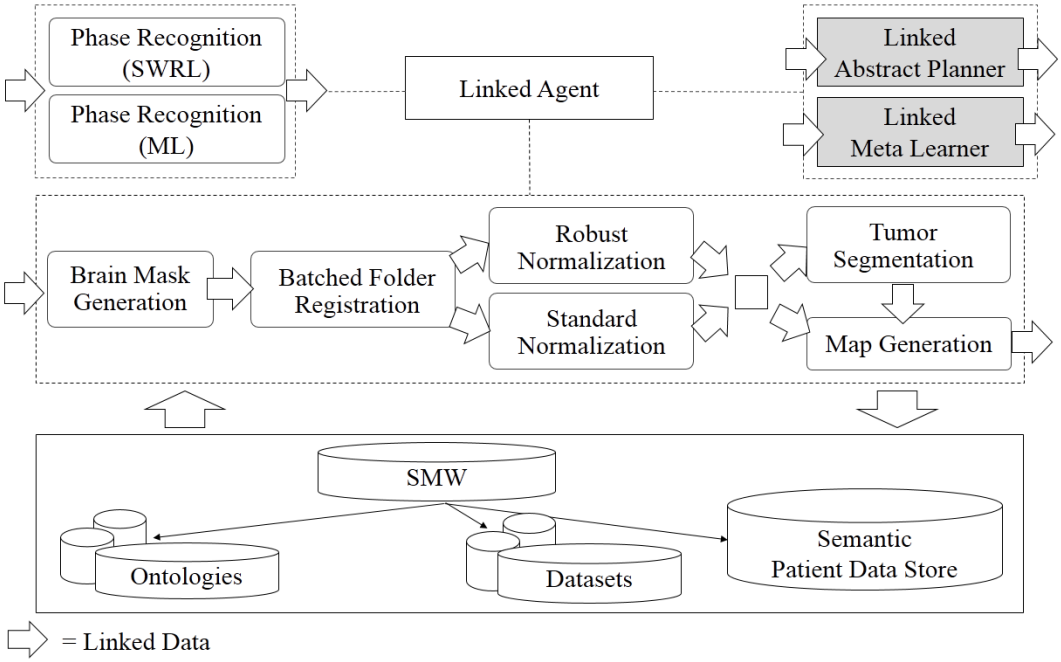

Fig. 6. The Semantic Framework for Medical Sequential Decision Making (extended based on [32]).

The TPM generation process is illustrated in the framework overview (fig.). The images are stored in our knowledge base and converted into a common format. A mask for the brain region is created by the next interpretation algorithm, ensuring that subsequent tasks are not influenced by bones or other structures. A registration algorithm, then, spatially registers all brain images of a patient. The following normalization task adapts the intensities of MRI scans and generates similar values for similar tissue types. If additional annotations for a patient are available, the normalization becomes more robust by making use of a different normalization interpretation algorithm. The TPM can now be created by invoking the appropriate interpretation algorithm. An optional additional interpretation algorithm can automatically segment tumors and integrate the results into the map.

We studied how to wrap interpretation algorithms used in the TPM setting in [17] and initially applied Linked Data-Fu in [33, 32]. We now integrate these ideas into the semantic framework and introduce a Linked planner.

5.2.1. *A Linked Abstract Planner for TPM.*

We developed Linked interpretation algorithms for every step in the TPM generation process. Listing 1 contains the preconditions of the brain stripping algorithm ('Brain Mask Generation') with headscan and initialization images as inputs.

```
?inputImage      rdf:type       kbont:Headscan;
                 dc:format      "image/nrrd".

?brainImage      rdf:type       kbont:BrainAtlasImage;
                 dc:format      "image/mha".

?brainAtlasMask  rdf:type       kbont:BrainAtlasMask;
                 dc:format      "image/mha".
```

Listing 1 Preconditions of the Linked brain mask generation algorithm.

We created a finite MDP by using the pre- and postconditions of Linked interpretation algorithms to define abstract states s_k with local scopes, i.e. we only consider preconditions of s_k. The transition probabilities T are defined in equation 1 and make up an $S \times (A+1) \times S$ matrix by adding a dummy interpretation algorithm pointing to the goal state, when the latter was reached. The reward function R is a $S \times (A+1)$ matrix (see equation 2). By using any strategy to solve the MDP (e.g value iteration), we find eligible Linked interpretation algorithms to solve the task.

$$T(s,a,s') = \begin{cases} t_{sas'} = \frac{1}{|A_{s_k}|} & \exists(s,a,s') \text{ with respect to } F_s \text{ and } F_{s'} \\ t_{sas'} = 0 & \text{otherwise} \end{cases} \tag{1}$$

$$R(s,a) = \begin{cases} r_{sa} = 1 & \text{if } a \text{ equals dummy algorithm and } s \text{ equals goal} \\ r_{sa} = 0 & \text{otherwise} \end{cases} \tag{2}$$

The resulting Linked abstract planner takes as input Linked interpretation algorithms, patient information of type **kbont:**ImageFeature (i.e. $g(s_k)$), and goal state **kbont:**TumorProgressionMapping (i.e. s_K), and returns a set of eligible Linked interpretation algorithms. The abstract Linked Data-Fu rule for the Linked abstract planner is depicted in listing 2. The Preconditions for the brain mask generation algorithm (see listing 1) could, then, replace the abstract image features.

```
{
?algo         rdf:type        kbont:InterprAlgorithm.
?goal         kbont:goal      kbont:TumorProgressionMapping.
?grounding    rdf:type        kbont:ImageFeature.
} => {
_:a   http:mthd         httpm:POST ;
      http:requestURI   kb:mplanermdp;
      http:body
    {
        ?algo         rdf:type        kbont:InterprAlgorithm.
        ?goal         kbont:goal      kbont:TumorProgressionMapping.
        ?grounding    rdf:type        kbont:ImageFeature.
    } .
} .
```

Listing 2 Linked Data-Fu rule for executing the Linked abstract planner.

Algorithm	Surgery 1	Surgery 2	Surgery 3	Surgery 4	Surgery 5
ML-based	0,9062	0,6635	0,9032	0,4484	0,6383
SWRL	0,9315	0,7753	0,89	0,8137	0,7241

Table 2. Performance evaluation of phase recognition algorithms in 5 different surgeries in terms of success rate.

5.2.2. *Evaluation.*

A part of the evaluation of the Linked TPM scenario was conducted in [32] and [17]. The TPM generation process was shown to work based on the descriptions of the single Linked image processing algorithms and an initial implementation of Linked Data-Fu without a Linked planner. We showed that no substantial overhead is produced while executing the interpretation algorithms on the web and that the correct pipeline is built automatically.

Our Linked abstract planner, now, creates a finite MDP and automatically constructs T and R based on the available Linked interpretation algorithms A, the goal s_K and the current grounding $g(s_k)$. Consider a grounding $g(s_k)$ for the brain stripping algorithm (listing 1) and the goal **kbont:**TumorProgressionMapping with all paths to the TPM being possible. Besides the 6 Linked interpretation algorithms involved in the TPM process, the algorithm pool consists of 2 Linked phase recognizers of the subsequent scenario. We use a discount factor of 0.9, perform value iteration and derive $V =< 0.32805.0.3645, 0.405.0.405, 0.45, 0.45, 1.00, 0, 0 >$ after 6 iterations. States with values greater than zero depict preconditions of Linked interpretation algorithms which have to be executed to reach the goal (except for the absorbing goal state s_K).

5.3. *Surgical Phase Recognition in the Semantic Framework*

Surgical phase recognition is crucial to reduce information overload for surgeons during surgery. Depending on the current phase, one can display an adequate subset of information, which benefits the surgeons in his or her decision making. To recognize the phase, one might leverage a variety of sensor outputs. In this scenario, only activity triples consisting of the currently used instrument, the performed action and the corresponding anatomical structure are used to determine the current phase (e.g. <Scalpel, cut, Gallbladder >).

The interpretation algorithms we considered for our learning scenario consisted of a rule-based interpretation algorithm using the Semantic Web Rule Language (SWRL) introduced in [22] and a machine learning (ML)-based phase recognition algorithm which needs to be trained with adequate samples (i.e. activity triples with the correct phase as label, retrieved from annotated video recorded surgeries). Both algorithms have varying degrees of performance and make mistakes in their predictions, as shown in table 2. If one could learn in which situations the respective interpretation algorithms excel, it would be highly beneficial. Our first approach to empirically learn the optimal phase recognition algorithm was mentioned in [32] and is now explained in terms of a generalizable Linked meta learner.

5.3.1. *A Linked Meta Learner for Surgical Phase Recognition.*

We developed Linked interpretation algorithms for both phase recognition algorithms, and defined their inputs and outputs in terms of semantic pre- and postconditions [32]. See listing 3 for the preconditions of the Linked ML-based phase recognition algorithm. The postcondition simply states that the result has to be of type **kbont:**Phase which ensures, by inference, that only modelled phases can occur. The resulting Linked interpretation algorithms need to be initialized with a laparoscopic ontology with concepts for the surgical setting. The ML-based phase recognizer, in addition, has to be trained with samples.

```
?trainingSample   rdf:type           kbont:Surgery.

?ontology         rdf:type           kbont:Ontology.

?event            rdf:type           kbont:SurgicalEvent;
                  kbont:instrument   ?instrument;
                  kbont:action       ?action;
                  kbont:structure    ?structure.

?instrument       rdf:type           kbont:Instrument.
?action           rdf:type           kbont:InstrumentalProperty.
?structure        rdf:type           kbont:TreatedStructure.
```

Listing 3 Preconditions of the Linked ML-based algorithm.

We developed a Linked meta learner for the setting of two competing Linked phase recognition algorithms. As it is quite specific and works only for Linked phase recognition algorithms, we define a less general description of the Linked meta learner. Listing 4 depicts the rule for executing the Linked meta learner. It assumes available candidate outputs of Linked interpretation algorithms to choose the best fitting surgical phase. Note that we can elegantly define the generality of the Linked meta learner based on the concept types we use. If it was able to optimally choose among two or more image processors as well, we could easily express that in the pre- and postconditions.

The Linked meta learning component assesses the performance of a given Linked phase recognizer based on training samples close to the current state $g(s_k)$. It trains the ML-based phase recognizer on parts of the training sets (i.e. all surgeries but one) and predicts on the remaining surgery. The process is repeated until we have predictions for all surgeries. $g(s_k)$ based on its performance on similar samples. Finally, the optimal hypothesis $V^{\star}(s')$, i.e. the value of the optimal grounding $g(s')$, is generated based on the weighted combination of available Linked interpretation algorithms a_i. The heuristic needs interpretation algorithms A, training sample states $S^{\texttt{Train}}$, similarity measures K, threshold function u for similarity measures, cut t for cross-validating machine learning-based interpretation algorithms and the current state $g(s)$ as inputs. It is summarized in algorithm 1.

```
{
?algo         rdf:type         kbont:PhaseRecognitionAlgorithm;
              kbont:result     ?result.
?mplaner      kbont:eligible   ?algo.
?grounding    rdf:type         kbont:StateFeature.
} => {
_:a   http:mthd        httpm:POST;
      http:requestURI  kb:mlearnerheuristic;
      http:body
    {
        ?algo        rdf:type      kbont:PhaseRecognitionAlgorithm;
                     kbont:result  ?result.
        ?grounding   rdf:type      kbont:StateFeature.
    } .
} .
```

Listing 4 Linked Data-Fu rule for executing the Linked meta learner.

Algorithm 1 $\texttt{MetaLearn}(A, S^{\texttt{Train}}, K, u, t, g(s))$

```
Train ← S^Train cut into t subsets
for all a ∈ A do
    if trainable(a) then
        PerformanceTable ← crossValidate(a, Train)
    else
        PerformanceTable ← validate(a, S^Train)
for all sim ∈ K do
    N^sim ← nearestNeighbours(g(s), sim, u(sim))
    for all a ∈ A do
        for all n^sim ∈ N^sim do
            R(n^sim, a) ← checkPerformance(a, n^sim, PerformanceTable)
    for all a ∈ A do
        V(s')^sim = V(g(s'))^sim + T(g(s), a, g(s'))R(g(s), a)
    V(g(s'))^sim⋆ ← arg max_{s'∈S} V(g(s'))^sim
for all g(s') ∈ g(S) do
    V(g(s')) ← (Σ_{sim∈K} V(g(s'))^sim⋆) / |K|
V(g(s'))⋆ ← arg max_{g(s')∈g(S)} V(g(s'))
return V(g(s'))⋆
```

5.3.2. *Evaluation.*

We determined the nearest neighbours based on a straightforward similarity measure for surgical activities, namely the exact pairwise match of structure, instrument and action. If for two activities the current triples plus their three predecessors match, the activities get assigned a similarity of 1. The similarity linearly decreases to 0.25 if only the current activity

Algorithm	Surgery 1	Surgery 2	Surgery 3	Surgery 4	Surgery 5
LMR	0,9332	0,7786	0,9180	0,7782	0,7238

Table 3. Performance evaluation of the Linked meta learner in terms of success rate [32].

triples match and has value 0 otherwise.

We performed five-fold cross validation, where we trained on four surgeries and predicted on the residual one. The evaluation metric we used was the success rate, which is defined as average of correct predictions for the test set.

The resulting success rate of the Linked meta learning component reached a better success rate than the best phase recognizer or was at least able to compete [32]. The results are summarized in table 3. In general, meta approaches learning a probability distribution over such algorithms often provide stabler results in the longterm, but often fail to choose the optimal algorithm for every single $g(s_k)$ in hindsight.

Based on meta learning for surgical phase recognition and abstract planning for tumor progression mapping, sequential decision-making can be enabled for medical scenarios. The central limitation for abstract planning is that no data point-specific choices are made for constructing the Linked interpretation algorithm pipelines and, as such, optimal results cannot be achieved. Similarly, we only explored meta learning for a one-step task, where the problem of considering future impacts when taking decisions is not prevalent. Finally, the need for developing grounded planning components did not occur, as pre- and postconditions were concisely defined.

We, now, deal with a NLP scenario, where all prior mentioned complexities are available. Here, the meta learning technique can be completely reused, but the semantic description for the respective Linked meta learners has to be adapted. However, novel Linked components for planning as well as planning-related learning have to be developed.

6. The Named Entity Recognition & -Disambiguation Scenario

The Web-related scenario focuses on the additional challenge of planning-related learning, i.e. learning the reward function R for more than one subtask x^k and choosing which transitions in T to follow.

In entity linking, one deals with the problem of ambiguity in unstructured text, mostly available on the Web. As a mention or surface form of an entity in text can be quite ambiguous, it often remains unclear which real world entity is actually being referred to. It is, thus, important to find links between such mentions and unique entities available in knowledge bases, such as Wikipedia [o], DBpedia [2] or Yago [47].

More specifically, entity linking consists of the subtasks named entity recognition (NER) and named entity disambiguation (NED). We will refer to the complete NERD task as x_{NERD} and will use x_{NER} and x_{NED} for the respective subtasks. In x_{NER}, one needs to annotate tokens in raw text as either being a distinct type of named entity (e.g. a person, a location or an organization) or not being a named entity (i.e. `NIL`). Other than in traditional NER, we only need a binary decision for solving x_{NERD}, i.e. x_{NER} only needs to map tokens (or

[o] https://en.wikipedia.org/wiki/Main_Page

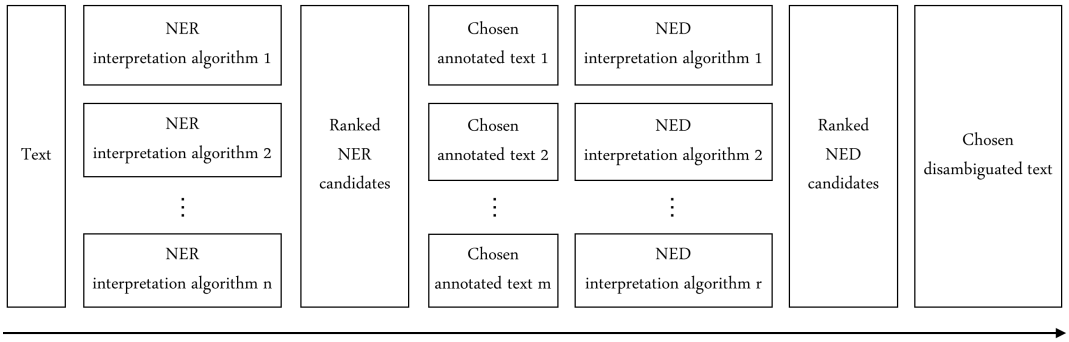

Fig. 7. Decision Process for x_{NERD}.

token sequences) to $[0, 1]$ with 1 referring to the presence of an entity and 0 to the absence. In this context, NER is often referred to as mention detection. x_{NED}, in turn, uses text annotated with named entity candidates as input and links the latter to a modelled resource in a structured (or semi-structured) knowledge base.

Although numerous interpretation algorithms for NERD have been developed so far – using diverse theoretical approaches for their predictions – no single one is paramount on all available data distributions on the Web, such as news articles or tweets. Tweets are short text snippets with a restriction to 140 characters and often contain highly colloquial and informally written speech, yielding different challenges than disambiguating news articles, which consist of longer text snippets with rather formal speech. Even if there was one NERD interpretation algorithm which dominated all others for all data sets, it would be highly unlikely that it was superior for all single articles or tweets. A combination of outputs of several NERD interpretation algorithms might, thus, yield even better performs than the single best NERD interpretation algorithm.

To achieve this, we explore meta-learning for x_{NERD}, which can be applied to x_{NER}, x_{NED} or x_{NERD}, i.e. we can combine NER interpretation algorithms, NED interpretation algorithms or interpretation algorithms which solve both x_{NER} and x_{NED} at once. In this paper, we will focus on combining NER- and NED interpretation algorithms separately. Our resulting decision process is visualized in figure 7.

As can be seen, retrieved candidate results are first combined for x_{NER}, but not only a single choice has to be used for approaching x_{NED}. Choosing annotations for x_{NED} is one part of the planning-related learning problem, as R (i.e. the reward of executing an interpretation algorithm) as well as T (i.e. the transitions we choose to use for x_{NED}) need to be estimated. While R is learned separately for x_{NER} and x_{NED}, and eventually averaged (see section refmetalearningsection), we are approaching estimating T by a sampling technique (see section). In addition, executing the NER- and NED interpretation algorithms entails novel challenges for a data-driven approach, which we tackle in section .

We will now present the planning-related learning approach for x_{NERD} and will subsequently evaluate its performance.

6.1. *A Planning-Related Learner for* x_{NERD}

We, first, present the semantic descriptions for NER and NED interpretation algorithms, as we need to develop Linked interpretation algorithms to automate the planning-related learning process. Based on the latter, we discuss our approach to estimate R and T for x_{NERD} based on available Linked interpretation algorithms for both x_{NER} and x_{NED}. The approach consists of Linked meta learners and a novel Linked planner to enable a data-driven, automatic and dynamic execution for solving x_{NERD}. We, therefore, reuse the Linked meta learner introduced in section and will present specifics of its application in the evaluation section.

6.2. *Semantic Descriptions for Linked NER & NED Interpretation Algorithms*

Numerous NER, NED and NERD interpretation algorithms are available as Web services and provide for rich descriptions of inputs and outputs (see section for examples). While inputs for x_{NER} are confined to text, outputs usually consist of mappings of tokens or token sequences to named entities with start and end positions of the latter, and the predicted type of the named entity (which we do not need for solving x_{NERD}). Depending on the implementation of the interpretation algorithm and the Web service, the output might also provide a confidence measure for each mapping. NED Web services, for the most part, need an annotated version of the text under consideration. The produced output consists of mappings from token or token combinations to knowledge base resources, the position of the token or token sequence and, if available, a confidence measure for each prediction. NER and NED Web services might provide further parameters which deal with their functionality.

For modelling the pre- and postconditions, we reuse the NLP Interchange Format (NIF) [p], a RDF vocabulary for describing both Web services and datasets for several NLP tasks. The preconditions for Linked NER algorithms are depicted in listing 5. It confines text inputs to be either sentences or complete paragraphs with adequate NIF annotations. Parameters specific to the respective interpretation algorithm can be set as well but remain optional as a default setting exists.

```
?text             rdf:type             ?textType.

FILTER (?textType = nif:Sentence || ?textType = nif:Paragraph).

OPTIONAL {
?parameterConfig  rdf:type             kbont:ParameterConfig;
                  kbont:parameter      ?parameter;
                  kbont:parameterValue ?parameterValue.
}
```

Listing 5 Preconditions of Linked NER algorithms.

The preconditions of Linked NED algorithms are depicted in listings 6. Inputs are NER annotations with information about start and end indexes of a named entity candidate (i.e. a token or a token sequence).

[p] http://persistence.uni-leipzig.org/nlp2rdf/

```
?annotation       rdf:type             kbont:Annotation;
                  kbont:text           ?textResource;
                  kbont:token          ?token.

?token            kbont:isEntity       "true";
                  nif:anchorOf         ?mention;
                  nif:beginindex       ?start;
                  nif:endIndex         ?end;
                  nif:referenceContext ?textResource.
OPTIONAL {
?parameterConfig  rdf:type             kbont:ParameterConfig;
                  kbont:parameter      ?parameter;
                  kbont:parameterValue ?parameterValue.
}
```

Listing 6 Preconditions of Linked NED algorithms.

Every NER and NED Linked interpretation algorithm considered in the scenarios was wrapped as Linked interpretation algorithm. We modelled the descriptions with NLP domain experts and developers of NER and NED interpretation algorithms, and made them available via the respectively developed Linked APIs. The texts and resulting annotations are stored in a triple store, which makes up the knowledge base together with the semantic description of the NER and NED Linked interpretation algorithms.

6.3. *Extending the Linked Meta Learner*

We reuse the Linked meta learner presented in section and instantiate it for both x_{NER} and x_{NED}. In contrast to the phase recognition scenario, we use multiple similarity measures, summarized in table 4.

Similarity Metric	NERD Application	Implementation
State length Length	Input text	Character length
State Extra Characters	’’, ’#’, smileys in input text	Count
Candidate Embeddings	Token- or named entities	Pretrained
Candidate MinHash	Token- or named entities	Jaccard similarity

Table 4. Used similarity measures for x_{NER} & x_{NED}.

Two similarity measures leverage text similarities to find nearest neighbours for the token candidates in x_{NER} and for the named entity candidates in x_{NED}. Here, we reuse existing implementations for word embeddings [q]and locality sensitive hashing [r]. The state dependent measures, i.e. text length and amount of extra characters, are calculated based on the input text snippets for both x_{NER} and x_{NED}.

One central difference to both the phase recognition and TPM scenario is the complexity of decision making per grounded state $g(s)$ for x_{NER} and x_{NED}. For x_{NER}, tokens or

[q]https://code.google.com/archive/p/word2vec/
[r]https://github.com/ekzhu/datasketch

token sequences of a text snippet (i.e. a grounded state $g(s)$) are mapped to named entity candidates at once, which clearly depends on $g(s)$, as the available amount of text potentially impacts the decision. The same holds for x_{NED} when mapping named entities to knowledge base resources or `NIL`. We, thus, define grounded decision candidates $c_s \in C_S$ as part of states $s \in S$ to express fine-grained decision making, dependent on the respective states. $T_C : C_S \times A \times C_S \rightarrow [0, 1]$ and $R_C : C_S \times A \rightarrow [0, 1]$ are defined accordingly, denoting the transition and reward functions for a decision candidate c_s.

The final values of disambiguated named entities $V(g(c_s))$ are calculated by averaging the scores retrieved by Linked meta learners for x_{NER} and x_{NED}, as formalized in algorithm 2

Algorithm 2 `CalculateV`$(c'_{s'})$

```
1: V ← 0
2: if k > 0 then
3:     for all a ∈ A do
4:         if T_C(c_s, a, c'_{s'}) = 1 then
5:             V = V + R_C(c_s, a)
6:             V = V + CalculateV(c_s)
7: else
8:     return 1
9: return V
```

6.4. *A Linked Planner for* x_{NERD}

In contrast to the TPM scenario, precondition groundings of Linked NED algorithms are not able to construe correct state configurations, as multiple token/named-entity mappings might have been chosen. To give an example, consider the following tweet:

Michael Jordan is a famous basketball player.

There is no ambiguity for x_{NER} and, thus, all available interpretation algorithms take the full text as input. Given the annotation candidates "Michael", "Jordan", "Michael Jordan" and "basketball player" one, now, has to construe valid annotations for x_{NED} which causes two problems:

1. The triple pattern of the NED precondition will never construe annotated texts with more than one annotation.
2. We cannot express that "Michael Jordan" and "Michael" must not be used for one text annotation due to their overlap.
3. It quickly becomes intractable to try out all valid annotations for x_{NED}.

We, therefore, develop a novel Linked planner and will first deal with problem 1, i.e. enabling complex grounding of preconditions to be created.

6.4.1. *Semi-Grounded Plans.*

We approach problem 1 by grounding responsible parts of the triple patterns while keeping the residuals variable such that the triple pattern returns eligible and correct solutions. However, we cannot ground NED inputs before having executed NER interpretation algorithms

and, thus, need to re-plan after every subtask x^k. Based on a selection of state configurations $g(s)$, the Linked Planner creates a new Linked Data-Fu program for the available NED interpretation algorithms. The procedure is formalized in algorithm 3.

Algorithm 3 `plan(g(S), A)`

```
1: program ← instantiate()
2: for all g(s) ∈ g(S) do
3:     for all a ∈ A do
4:         rule ← semiGroundPrecondition(g(s), a)
5:         program ← program ∪ rule
6: return program
```

The algorithm can be easily extended to the case where not all candidate solutions in $g(S)$ should be executed by every Linked interpretation algorithm. This might be needed in budgeted scenarios where meta learners have to abide by constraints.

Problems 2 and 3 – generating a reduced set of valid candidates for the Linked planner – are approached by a sampling technique which will be discussed next (section).

6.4.2. *A Sampler for Valid State Configurations.*

Trying out all NER output combinations quickly becomes intractable, as some Linked NER algorithms potentially produce a large set of outputs (i.e. candidates $c_s \in C_s$). The problem becomes more serious with more subtasks, as might be the case for the TPM use case we dealt with in section . An upper bound to the number of executions is defined by all possible n-grams (with $n = 1, \ldots, |\texttt{tokens(text)}|$) of a text.

After several Linked NER algorithms haven been executed and a Linked meta learner has ranked their outputs, we have to restrict the number of configurations ($g(S)$) we want the Linked planner to build. We propose a sampling approach which generates probability distributions based on the weights of the Linked meta learner and draws a predefined number of samples. We, therefore, iteratively cluster all named entity candidates $c_s \in C_s$ until we derive valid named entity assignments for a text $g(s)$. More specifically, we first create clusters for each named entity candidate c_s to decide if we consider it for a sampling round. As there might be several conflicting named entity assignments, we use the retrieved samples to create new clusters for each sampled candidate c_s plus its dependent named entity candidates (i.e. the candidates which overlap c_s in $g(s)$) The sampling process continues until a valid annotation is reached, i.e. we have mapping from tokens to named entities which do not overlap. The procedure is formalized and summarized in algorithm 4, having the set of candidate named entities C_s and their prior computed values V as input.

6.5. *Evaluation*

For our evaluation, we implemented the novel Linked planner which semi-grounds preconditions of Linked interpretation algorithms based on sampled state configurations. We, also, implemented Linked meta learners for x_{NER} and x_{NED} based on the Linked meta learner for surgical phase recognition and compute the final candidate values $V(c)$ based on algorithm 2. The performance of our approach is quantified in terms of correct mappings from tokens

Algorithm 4 `sample`(C_s, V)

```
clusters ← ∅
for all c_s ∈ C_s do
    clusters ← clusters ∪ (c_s, ¬c_s)
divide ← true
while divide do
    for all cluster ∈ clusters do
        P ← getProbabilityDistribution(c_s)
        clusters.decision ← draw(P) // 1 if named entity
    divide ← checkValidity(clusters) // false if assignment is valid
    if divide then
        C_Dep = neighbourDependencies(clusters)
        clusters ← ∅
        for all c ∈ C_Dep do
            clusters = clusters ∪ (c ∪ c.deps) // c.deps returns the dependencies
g(S) ← transformToStates(clusters)
return g(S)
```

and token sequences of a text to disambiguated named entities.

6.5.1. *Data.*

Our approach is applied to two datasets, namely Microposts 2014 train+test [3] and Spotlight Corpus [28]. As the predictive performance of available interpretation algorithms significantly varies for tweets, applying a planning-related learning approach might both improve predictions and make them more robust. In addition, with limited amounts of labeled data for benchmarking interpretation algorithms, knowledge transfer becomes important. We, thus, evaluate our approach in terms of reusing tweets for solving x_{NERD} for articles.

6.5.2. *Baseline.*

The baseline to our approach consists of 1) individual performances of the used interpretation algorithms and 2) static weighted combination of the individual interpretation algorithms. For computing the former we use GERBIL [50], a benchmark for entity linking which features well known NERD interpretation algorithms and numerous data sets to quantify their performance. For baseline 2) we implemented an additional Linked meta learner which uses the individual f1-measure scores of a dataset as static weights.

6.5.3. *Setup.*

Table 5 summarizes the NER and NED interpretation algorithms we used. We only used implementations available as Web services and did not tune parameters, i.e. we took the default settings the Web services provided.

The results for baseline 1) are available via GERBIL [s]Here, the NER- and NED interpretation algorithms of table 5 are merged to their original NERD setting. That is, DBpedia

[s] http://gerbil.aksw.org/gerbil/experiment?id=201605010000.

Interpretation algorithm	Task
Stanford Tagger [15]	NER
FOX [42]	NER
Spotlight Spotter	NER
AGDISTIS [49]	NED
AIDA [20]	NED
Spotlight Tagger [28]	NED

Table 5. NER and NED interpretation algorithms.

Spotlight Spotter and -Tagger make up DBpedia Spotlight, AGDISTIS uses FOX as NER approach and AIDA relies on Stanford Tagger to propose named entities. Note that the results for baseline 1) in table 6 deviate from the ones by GERBIL, as we needed to re-evaluate the interpretation algorithms for the Spotlight corpus. Here, we treated every sentence as single grounded state $g(s)$. Baseline 2) uses, as already mentioned, static weights to combine NER and NED interpretation algorithms (see "Static weights" in the result table). The approach was proposed by [38] in a slightly different version, additionally incorporating a rank feature for interpretation algorithms. Our approach, `LinkedPRL`, uses 1800 tweets of the Microposts 2014 averaged dataset as training data for solving x_{NERD} for tweets *and* articles. We, thus, also evaluate its ability of knowledge transfer, as only tweets are used for training when predicting disambiguated named entities for the Spotlight corpus. For both instantiations, we sample 8 valid state configurations for x_{NED}. The results were retrieved by 10-fold cross validation for learning and predicting on the same dataset. We evaluate the approaches in terms of precision (Prec), recall (Rec) and f1-measure (F1). Note that we only make relative statements about the performance of our approach as it is dependent on the residual interpretation algorithms.

6.5.4. *Results.*

Table 6 summarizes our results.

	Microposts 2014 averaged			Spotlight		
Approach	F1	Prec	Rec	F1	Prec	Rec
Spotlight	0.4887	0.6666	0.3859	0.419	0.3598	**0.5013**
AGDISTIS	0.3713	0.37135	0.3713	0.1795	0.5526	0.1071
AIDA	0.379	0.5388	0.2924	0.2437	**0.6905**	0.148
Static weights	0.4025	0.5036	0.3352	0.3598	0.477	0.2889
`LinkedPRL`	**0.5143**	**0.7351**	**0.3955**	**0.4213**	0.4977	0.3652

Table 6. Evaluation results for x_{NERD}.

In total, our Linked planning-related learner yields superior results for tweets compared to the individual interpretation algorithms and the statically weighted combination approach. For articles, DBpedia Spotlight and AIDA are dominating in terms of precision and recall. `LinkedPRL`, however, provides stable results and yields the best f1-measure score and, thus, is able to transfer knowledge from tweets to articles.

Based on our qualitative and quantitative findings for developing and integrating Linked interpretation algorithms as well as Linked meta components for the Semantic Framework, we now discuss central questions concerning its generalizability and limitations.

7. Discussion

We split our discussion into (i) the added value of the framework in general, (ii) the generalizablity of Linked meta components as well as Linked interpretation algorithms to new domains or new tasks, (iii) the problem of different reward impacts and (iv) the overall goal of our approach.

7.1. *On the Added Value of the Semantic Framework*

Modelling fine-grained semantic annotations, and developing and integrating meta components adds manual work for domain- and machine learning experts. Still, resulting Linked meta components enable to gradually learn which Linked interpretation algorithms to select given characteristics of data points and how to automatically pipeline them. Pipelines can thus be dynamically changed for each data point to reach better performances. The latter would otherwise remain static unless manually changed for each single data point or unless fine-grained preconditions are available which have to be frequently updated.

Also, automatically taking into account novel Linked interpretation algorithms with adequate semantic interfaces is not trivial without the Semantic Framework. As a consequence, instantiated workflow systems for a task often remain unchanged irregardless of novel candidate interpretation algorithms being available.

7.2. *On the Generalizability of Linked Components to New Domains or Tasks*

The ability to generalize and reuse the Linked meta components we developed throughout this work is dependent on (i) the flexibility of the underlying meta component and (ii) its semantic description. (i) Our meta learner is based on the idea of instance-based learning and only requires domain-dependent similarity metrics to be defined for new data points. Our meta planer is based on MDPs and completely generalizable. (ii) The semantic descriptions we modelled are domain-dependent for both meta learning and meta planning, as they need to capture respective constraints for execution and correct functioning. The semantic descriptions for Linked meta learners can, however, be automatically inferred from Linked interpretation algorithms, while Linked meta planners need to capture goal-dependent task parameters, such as the constraint of having one decision per token or token sequence for NERD.

To this end, one could relax the impact of the pre- and postconditions and shift the decision to planning-related learning components, i.e. model generic pre- and postconditions and learn their eligibility for a datapoint via feedback. This might be useful for generalizing the applicability of a Linked interpretation algorithm or for dealing with situations where few semantics are available.

Also, using the framework for novel tasks is dependent on the goal of the respective task, as Linked meta components as well as Linked interpretation algorithms have to be modelled accordingly. More specifically, if a novel task could theoretically be solved by available Linked

interpretation algorithms in terms of their functionality, their semantic description might not capture all needed information of the task's goal. To this end, the respective Linked API might have to be adjusted, as extended semantic annotations might influence the underlying interpretation algorithm.

7.3. *On Domain-dependent Impacts of Rewards*

The meta learning and planning-related learning tasks both entail learning R, the reward function for interpretation algorithms. We neglected the aspect that R might have different interpretations and impacts for different domains. While for NLP techniques applied to Web-based tasks, choosing the highest estimated reward measured on past executions might not cause problems, it might have critical and ethical consequences for medical scenarios. Provenance metadata might have to be extended with more specific criteria for decision-making, where manually defined decision policies potentially need to be followed based on strict medical guidelines. Hence, a static interpretation algorithm pipeline might be preferred for all data points, as it is trusted by physicians.

A related challenge is that institutions might differ in their perceptions of what conceptual provenance metadata is needed to trust it or that different guidelines are followed. Hence, reusing task outcomes published as Linked Data requires developing strategies to deal with uncertainties coming from missing information.

7.4. *On The Goal of an Self-Adaptive System*

Since we enable to use multiple Linked meta components at once, one could have them compete as well. This is what Vilalta & Drissi [52] depict as curse of infinite bias. We want the system to be self-adaptive and improve with experience, which it already does to some extent by automatically considering training samples or further data sources. However, each Linked meta components has some kind of bias in terms of their methodology used. Hence, while is might interesting to have meta components compete with each other, the problem how to deal with their respective bias is important on its own.

8. Conclusion

We introduced our work on a proposing a first semantic framework for sequential decision making in a heterogeneous environment. We reused established techniques of the Semantic Web to develop a data-driven, declarative framework for Linked interpretation algorithms and extended it with means to solve complex tasks. Therefore, the problem of complex task solving was defined and we distinguished between (abstract-) planning- and (meta-) learning scenarios, with initial observations on the interplay with Linked Data. By now, we realized solutions to all prior defined problems with first Linked meta components and Linked interpretation algorithms, which can be naturally integrated with the Linked agent (contribution (i)). We applied the framework to two exemplary use cases in the medical domain and to one use case in the Web domain. All interpretation algorithms – i.e. image processors, phase recognizers, and named entity recognizers -and disambiguators – were transformed into Linked interpretation algorithms by wrapping them as Linked APIs and executed by the Linked agent. The Linked meta components we developed for the use cases realized and optimized the pipeline

construction for solving our complex tasks (contributions (ii) and (iii)). We, finally, discussed current shortcomings of our framework, potential improvements we are investigating and the long-term goal of a self-adaptive framework.

9. Future Work

In case of abstract planning and MDPs, we only leveraged semantics to a small degree, namely in terms of pre- and postcondition matchings. We want to investigate the potential advantages of richer classes such as relational MDPs [31]. In addition, linked meta components can make large use of an arbitrary amount of features besides their preconditions. Linked interpretation algorithm descriptions, although potentially modelled by domain experts, do not necessarily capture all relevant dependencies. Hence, learning the optimal feature subset of F_{s_k} to better estimate transition probabilities $T(s, a, s')$ or enriching F_{s_k} with more features seem interesting extensions to our framework. We also want to develop new Linked meta components for the pure planning and planning-related learning task as defined in section .

The Linked meta learner we developed makes fine-grained predictions by using training data of instance-specific neighbourhoods. Therefore, numerous heterogeneous similarity measures can be easily incorporated to define these neighbourhoods. However, resulting predictions are evenly averaged and no weights for the individual impact of similarity measures are used. Thus, one extension of our meta learner would be approaching to learn a supervised model to dynamically weigh the results proposed in different neighbourhoods.

While we already generate provenance metadata based on Linked interpretation algorithm as well as Linked meta component executions, reusing established sophisticated ontologies (such as OPMW) is beneficial and might open further opportunities and research challenges. One example might be discovering and selecting interpretation algorithms when no training data is available, which requires to examine generated metadata from other institutions (published as Linked Data). To this end, guaranteeing reproducibility of results and determining if sufficient contextual information is available to trust these outcomes are essential.

Finally, our evaluations showed that the proposed Linked Agent using Linked meta components is able to pipeline Linked interpretation algorithms for medical and Web-related tasks, thereby achieving high outcome performances. However, with regards to prior discussed critical reward impacts, further qualitative and quantitative evaluations are needed to work towards richer provenance models and thus higher end user acceptance of automatically pipelined workflows, e.g. end user questionnaires to quantify trust thresholds for suggested task solutions.

Acknowledgements.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 611346 and from the German Research Foundation (DFG) for the SFB/Transregio 125 "Cognition-Guided Surgery". We especially thank Stefanie Speidel, Christian Weber, Michael Götz, Anna-Laura Wekerle, Miriam Klauß, Hannes Kenngott and Beat Müller-Stich for support with the medical use cases.

References.

1. Albrecht, M., Donnelly, P., Bui, P., Thain, D.: Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids. In: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies. pp. 1:1–1:13. SWEET '12, ACM (2012)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. pp. 722–735 (2007)
3. Basave, A.E.C., Rizzo, G., Varga, A., Rowe, M., Stankovic, M., Dadzie, A.: Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In: Proceedings of the 4th Workshop on Making Sense of Microposts co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 7th, 2014. pp. 54–60 (2014)
4. Benjamins, V.R., Pierret-Golbreich, C.: Assumptions of problem-solving methods. In: European Knowledge Acquisition Workshop, EKAW. pp. 1–16 (1996)
5. Beygelzimer, A., Riabov, A., Sow, D., Turaga, D.S., Udrea, O.: Big data exploration via automated orchestration of analytic workflows. In: ICAC. pp. 153–158. San Jose, CA (2013)
6. Blankenberg, D., Kuster, G.V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A., Taylor, J.: Galaxy: a web-based genome analysis tool for experimentalists. Current protocols in molecular biology pp. 19–10 (2010)
7. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al.: Web services description language (wsdl) 1.1 (2001)
8. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P., Mayani, R., Chen, W., da Silva, R.F., Livny, M., Wenger, R.K.: Pegasus, a workflow management system for science automation. Future Generation Comp. Syst. 46, 17–35 (2015)
9. Doshi, P., Goodwin, R., Akkiraju, R., Verma, K.: Dynamic workflow composition using markov decision processes. In: ICWS. pp. 576–582 (2004)
10. Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.L., Villazon, A., Wieczorek, M.: ASKALON: A Development and Grid Computing Environment for Scientific Workflows, pp. 450–471. Springer London (2007)
11. Fensel, D., Motta, E., van Harmelen, F., Benjamins, V.R., Crubézy, M., Decker, S., Gaspari, M., Groenboom, R., Grosso, W.E., Musen, M.A., Plaza, E., Schreiber, G., Studer, R., Wielinga, B.J.: The unified problem-solving method development language UPML. Knowl. Inf. Syst. 5(1), 83–131 (2003)
12. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: NIPS. pp. 2962–2970 (2015)
13. Fikes, R., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artif. Intell. 2(3/4), 189–208 (1971)
14. Filguiera, R., Klampanos, I., Krause, A., David, M., Moreno, A., Atkinson, M.: dispel4py: A python framework for data-intensive scientific computing. In: Proceedings of the 2014 International Workshop on Data Intensive Scalable Computing Systems. pp. 9–16. DISCS '14, IEEE Press (2014)
15. Finkel, J.R., Grenager, T., Manning, C.D.: Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA (2005)
16. Garijo, D., Gil, Y.: A new approach for publishing workflows: abstractions, standards, and linked data. In: WORKS. pp. 47–56 (2011)
17. 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)
18. Gil, Y., Gonzalez-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)
19. Hendler, J.A., Tate, A., Drummond, M.: AI planning: Systems and techniques. AI Magazine 11(2), 61–77 (1990)
20. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 782–792 (2011)
21. Jain, A., Ong, S.P., Chen, W., Medasani, B., Qu, X., Kocher, M., Brafman, M., Petretto, G., Rignanese, G., Hautier, G., Gunter, D.K., Persson, K.A.: Fireworks: a dynamic workflow system designed for high-throughput applications. Concurrency and Computation: Practice and Experience 27(17), 5037–5059 (2015)
22. Katic, D., Wekerle, A.L., Gärtner, F., Kenngott, H.G., Müller-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. 158–167 (2014)
23. Klusch, M., Gerber, A., Schmidt, M.: Semantic web service composition planning with owls-xplan. In: Int. AAAI Fall Symposium on Agents and the Semantic Web. pp. 55–62 (2005)
24. Klusch, M.: Semantic web service coordination. CASCOM: Intelligent service coordination in the semantic web pp. 59–104 (2008)
25. Kopecky, J., Gomadam, K., Vitvar, T.: hrests: An html microformat for describing restful web services. In: Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on. vol. 1, pp. 619–625. IEEE (2008)
26. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: SAWSDL: semantic annotations for WSDL and XML schema. IEEE Internet Computing 11(6), 60–67 (2007)
27. Martin, D.L., Burstein, M.H., McDermott, D.V., McIlraith, S.A., Paolucci, M., Sycara, K.P., McGuinness, D.L., Sirin, E., Srinivasan, N.: Bringing semantics to web services with OWL-S. World Wide Web 10(3), 243–277 (2007)
28. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011. pp. 1–8 (2011)
29. Nguyen, P., Hilario, M., Kalousis, A.: Using meta-mining to support data mining workflow planning and optimization. J. Artif. Intell. Res. (JAIR) 51, 605–644 (2014)
30. 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)
31. van Otterlo, M.: The logic of adaptive behavior: knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains, vol. 192. Ios Press (2009)
32. Philipp, P., Katic, D., Maleshkova, M., Rettinger, A., Speidel, S., Wekerle, A.L., Kämpgen, B., Kenngott, H., Studer, R., Dillmann, R., Müller, B.: Towards cognitive pipelines of medical assistance algorithms. In: Proc. Computer Assisted Radiology and Surgery (CARS) (2015)
33. 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 fÃijr die Medizin (BVM). pp. 263–268 (2015)
34. Puppe, F.: Systematic introduction to expert systems - knowledge representations and problem-solving methods. Springer (1993)
35. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (1994)
36. Rice, J.R.: The algorithm selection problem. Advances in Computers 15, 65–118 (1976)
37. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C.,

Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology 1(1), 77–106 (2005)
38. Ruiz, P., Poibeau, T.: Combining open source annotators for entity linking through weighted voting. In: Proceedings of* SEM 2015. Fourth Joint Conference on Lexical and Computational Semantics (2015)
39. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education (2010)
40. Sirin, E., Hendler, J.A., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: WSMAI. pp. 17–24 (2003)
41. Sirin, E., Parsia, B., Wu, D., Hendler, J.A., Nau, D.S.: HTN planning for web service composition using SHOP2. J. Web Sem. 1(4), 377–396 (2004)
42. Speck, R., Ngomo, A.N.: Ensemble learning for named entity recognition. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. pp. 519–534 (2014)
43. 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)
44. Stadtmüller, S., Norton, B.: Scalable discovery of linked apis. IJMSO 8(2), 95–105 (2013)
45. Stadtmüller, S., Speiser, S., Harth, A., Studer, R.: Data-fu: A language and an interpreter for interaction with read/write linked data. In: WWW. pp. 1225–1236 (2013)
46. Strehl, A.L., Littman, M.L.: Online linear regression and its application to model-based reinforcement learning. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007. pp. 1417–1424 (2007)
47. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. pp. 697–706 (2007)
48. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: SIGKDD. pp. 847–855 (2013)
49. Usbeck, R., Ngomo, A.N., Röder, M., Gerber, D., Coelho, S.A., Auer, S., Both, A.: AGDISTIS - graph-based disambiguation of named entities using linked data. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. pp. 457–471 (2014)
50. Usbeck, R., Röder, M., Ngomo, A.N., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, J., Wesemann, L.: GERBIL: general entity annotator benchmarking framework. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015. pp. 1133–1143 (2015)
51. Verborgh, R., Harth, A., Maleshkova, M., Stadtmüller, S., Steiner, T., Taheriyan, M., Van de Walle, R.: Survey of Semantic Description of REST APIs, pp. 69–89. Springer New York (2014)
52. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review 18(2), 77–95 (2002)
53. Vitvar, T., Kopecký, J., Viskova, J., Fensel, D.: Wsmo-lite annotations for web services. In: ESWC. pp. 674–689 (2008)
54. Wood, I., Vandervalk, B., McCarthy, L., Wilkinson, M.: Owl-dl domain-models as abstract workflows. In: 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)