# An Incremental Learning Method to Support the Annotation of Workflows with Data-to-Data Relations

Conference or Workshop Item

# An incremental learning method to support the annotation of workflows with data-to-data relations

Enrico Daga[1], Mathieu d'Aquin[1], Aldo Gangemi[2,3], and Enrico Motta[1]

[1] Knowledge Media Institute (KMI) - The Open University.
Walton Hall, MK76AA Milton Keynes, United Kingdom
{enrico.daga,mathieu.daquin,enrico.motta}@open.ac.uk
[2] National Research Council (CNR), Via Gaifami 18, 95126 Catania, Italy
[3] Paris Nord University, Sorbonne Cite CNRS UMR7030, France
aldo.gangemi@cnr.it

**Abstract.** Workflow formalisations are often focused on the representation of a process with the primary objective to support execution. However, there are scenarios where what needs to be represented is the effect of the process on the data artefacts involved, for example when reasoning over the corresponding data policies. This can be achieved by annotating the workflow with the semantic relations that occur between these data artefacts. However, manually producing such annotations is difficult and time consuming. In this paper we introduce a method based on recommendations to support users in this task. Our approach is centred on an incremental rule association mining technique that allows to compensate the cold start problem due to the lack of a training set of annotated workflows. We discuss the implementation of a tool relying on this approach and how its application on an existing repository of workflows effectively enable the generation of such annotations.

## 1 Introduction

Research in workflows has been characterized on a variety of aspects, spanning from representation and management to preservation, reproducibility, and analysis of process executions [16, 11, 13, 14, 17]. Recently, a data-centric approach for the representation of data relying systems has been proposed with the aim to simulate the impact of process executions on the data involved, particularly to perform reasoning on the propagation of data policies [6, 4]. This approach puts the data objects as first class citizens, aiming to represent the possible semantic relations among the data involved. Annotating data intensive workflows is problematic for various reasons: (a) annotation is time consuming and it is of primary importance to support the users in such activity, and (b) workflow descriptions are centred on the processes performed and not on the data, meaning that some form of remodelling of the workflow is required. In this paper we introduce a method based on recommendations to support users in producing data-centric annotations of workflows. Our approach is centred on an incremental rule association mining technique that allows to compensate the cold start problem due to the lack of a training set of annotated workflows. We discuss the implementation of a tool relying on this approach and how its application on an existing repository of workflows (the "My experiment"[4] repository) effectively enables the generation of such annotations. In the next Section we introduce the related work. Section 3 describes the approach and Section 4 how it has been implemented in a tool that allows to annotate workflows as data-centric descriptions. In Section 5 we present the results of an experiment performed with real users where we measured how this method impacts the sustainability of the task. Finally, we discuss some open challenges and derive some conclusions in the final Section 6.

## 2 Related Work

In this paper we introduce a novel approach to recommend (semantic, data-centric) annotations for workflows. Research on process formalization and description covers a variety of aspects, from the problem of reproducibility to the ones of validation, preservation, tracing and decay [20, 7, 11, 22, 3]. Several models have been proposed for describing workflow executions, like the W3C PROV Model[5], the Provenance Model for Workflows (OPMW)[6] and more recently the Publishing Workflow Ontology (PWO)[7] introduced in [9]. A recent line of research is focused on understanding the activities behind processes in workflows, with the primary objective to support preservation and reusability of workflow components, particularly in the context of scientific workflows [2, 10]. We place our work in the area of semantic annotation of workflows. Semantic technologies have been used in the past to analyze the components of workflows, for example to extract common structural patterns [8]. Recently more attention has been given to the elicitation of the activity of workflows in a knowledge principled way, for example searching for common motifs in scientific workflows [10] or labelling data artifacts to produce high level execution traces (provenance) [1]. This research highlighted the need for adding semantics to the representation of workflows and the challenges associated with the problem of producing such annotations [1]. Recently a number of repositories of scientific workflows have been published - Wings[8], My experiments[9], SHIWA[10] are the prominent examples. We selected the My experiments repository as data source for our study. For this reason, we will use the terminology of the SCUFL2 model[11] when discussing how our approach deals with the workflow formalization.

[4] My experiment: `http://www.myexperiment.org/`.

[5] W3C PROV: `https://www.w3.org/TR/prov-overview/`.

[6] OPMW: `http://www.opmw.org/`.

[7] PWO: `http://purl.org/spar/pwo`.

[8] Wings: `http://www.wings-workflows.org/`.

[9] My experiments: `http://www.myexperiment.org/`.

[10] SHIWA: `http://www.shiwa-workflow.eu/wiki/-/wiki/Main/SHIWA+Repository`

[11] SCUFL2: `https://taverna.incubator.apache.org/documentation/scufl2/`.

There are several approaches to recommendation using clustering techniques (Support Vector Machines (SVM), Latent Semantic Aanalysis (LSA), to name a few). Formal Concept Analysis (FCA) [21] found a large variety of applications [19], and the literature reports several approaches to incremental lattice construction [15], including the Godin [12] algorithm, used in the present work. FCA found application in knowledge discovery as a valuable approach to association rule mining (ARM) [19]. In the context of FCA, association rules are generated from closed item sets, where the association rule to be produced relates attributes appearing in the intent of the same concept. A large number of studies focused on how to reduce the number of item sets to explore in order to obtain a complete set of minimal rules [19]. In the scenario of the present study, where the lattice changes incrementally, generating all the possible association rules would be a waste of resources. The algorithm proposed in the present work is *on demand*, as it only extracts the rules that are relevant for the item to annotate. Our algorithm receives as input an item set, and *retrieves* from the lattice the association rules associated with a relevance score. In other words, we follow an approach unusual with respect to the literature, attacking the ARM problem as an Information Retrieval (IR) one.

The approach presented in this paper uses the Datanode ontology [5], a hierarchy of possible relations between data objects. The ontology defines a unique type - *Datanode* - and 114 relations, starting from a single top property: *relatedWith*, having the class *Datanode* as `rdfs:domain` and `rdfs:range`. Datanode relations can express meta-level aspects (e.g. *describes/describedBy*, *hasAnnotation/isAnnotationOf*), containment (e.g. *hasPart/isPartOf*, *hasSection/isSectionOf*) as well as a properties like derivation (e.g. *hasCopy/isCopyOf*, *processedInto/processedFrom*), among others. Relations are organised by the means of the `rdfs:subPropertyOf` property. For example, *processedInto* is a subproperty of *hasDerivation*, as it is possible to derive a new data object from another also in other ways, for example generating an unprocessed copy - *hasCopy*. In the present work, a *datanode* is any data object that can be the input or output of a workflow processor. Instead on characterizing the activities of a workflow (like in [10]), Datanode can be applied to describe it in terms of relations between the input and the output of processors[12]. The resulting network of data objects can be used to reason upon the propagation of policies, for example in the context of a Smart City data hub [6, 4].

## 3 Recommendations for data-centric workflow annotations

Our approach to the problem is an iterative supervised annotation process supported by incremental recommendations. Figure 1 provides an overview of the approach by listing the elements and their dependency, organised in four phases. **Phase 1**. The starting point is an encoded artefact representing the workflow structure and its metadata (like the ones available through My experiments).

[12] Datanode: `http://purl.org/datanode/ns/`.

The workflow code is first translated into a data centric graph, where nodes are data objects manipulated by processors and arcs the relations among them. The result of this transformation is a directed graph with anonymous arcs (named IO port pairs in the Figure), being these arcs the items to be annotated by the user.
**Phase 2**. Each IO port pair is then associated with a set of features automatically extracted from the workflow metadata.
**Phase 3**. Extracted features constitute the input of the recommendation engine, designed using the Formal Concept Analysis (FCA) framework. This method is an incremental association rules mining technique that exploits incoming annotations to incrementally produce better recommendations.
**Phase 4**. Features of the IO port pair, alongside the workflow documentation and the recommendations, are the input of the user that is requested to select a set of annotations from a fixed vocabulary (the Datanode ontology).

In this section we focus on the first three phases of the approach: the workflow to data graph transformation (Section 3.1); the features extraction method (Section 3.2); and the recommendation engine (Section 3.3), leaving the last one to Section 4.

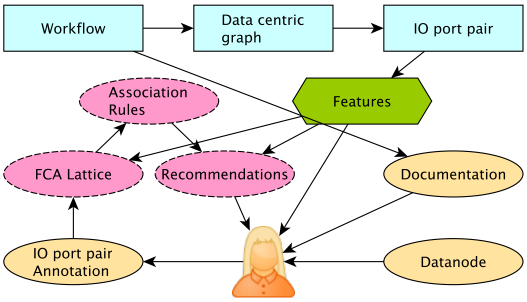

**Fig. 1.** Description of the approach and dependencies. Elements of phase 1 are represented in blue rectangles on top. Phase 2 includes the features generation (the only stretched exagon). Elements of Phase 3 are depicted as pink ovals with dashed borders and phase 4 ones as light yellow ovals.

### 3.1 Workflows as data-centric graphs

Workflows are built on the concept of *processor* as unit of operation[13]. A *processor* constitutes of one or more input and output *ports*, and a specification of the operation to be performed. Processors are then linked to each other through a set of data links connecting an output port to the input of another proces-

[13] In thi paper we use the terminology of the SCUFL2 specification. However, the basic structure is a common one. In the W3C PROV-O model this concept maps to the class *Activity*, in PWO with *Step*, and in OPMW to *WorkflowExecutionProcess*, just to mention few examples.

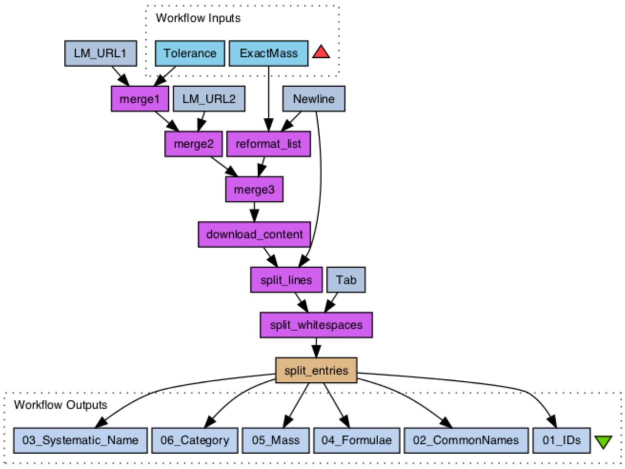

**Fig. 2.** A workflow from the My Experiment repository: "LipidMaps Query".

sor resulting in a composite tree-like structure. Figure 2 shows an example of a workflow taken from the "My Experiment" repository [14].

The objective of our work is to describe what happens inside the processors by expressing the relation between *input* and *output*. For example, the processor depicted in Figure 3 has two input ports (1 and 2) and one output (3). For this processor, we generate two links connecting the input data objects to the output one, through two anonymous arcs: $1 \rightarrow 3$ and $2 \rightarrow 3$. We name these arcs "IO port pairs" (input-output port pairs), and these are the items we want to be annotated. In this example, the IO port pair $1 \rightarrow 3$ could be annotated with the Datanode relation $refactoredInto$, while the IO port pair $2 \rightarrow 3$ would not be annotated as only referring to a configuration parameter of the processor and not to an actual data input. For the present work we translated 1234 Workflows from the My Experiments repository, resulting in 30612 IO port pairs (altough we will use a subset of them in the user evaluation).

### 3.2 Extracting features from workflow descriptions

As described in the previous Section, the workflow description is translated in a graph of IO port pairs connected by unlabelled links. In order to characterize the IO port pair we exploit the metadata associated with the components of the workflow involved: the input and output port and the processor that includes them. For each of these elements we extract the related metadata as key/value pairs, which we use as core features of the IO port pair. Applying this approach to the My Experiments corpus we obtained 26900 features. Table 1 shows an example of features extracted for the IO port pairs described in Figure 3.

[14] "LipidMaps Query" workflow from My experiment: `http://www.myexperiment.org/workflows/1052.html`.

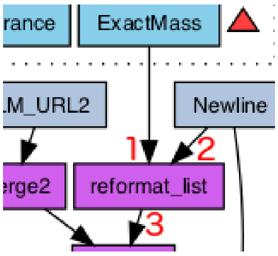

**Fig. 3.** This processor has three ports: two input ports (1 and 2) and one output port (3). We can translate this model into a graph connecting the data objects of the inputs to the one of the output.

**Table 1.** Sample of the features extracted for the IO port pair 1 → 3 in the example of Figure 3.

| **Type** | **Value** |
|---|---|
| From/FromPortName | string |
| To/ToPortName | split |
| Activity/ActivityConfField | script |
| Activity/ActivityType | `http://ns.taverna.org.uk/2010/activity/beanshell` |
| Activity/ActivityName | reformat_list |
| Activity/ConfField/derivedFrom | `http://ns.taverna.org.uk/2010/activity/localworker/org.embl.ebi.escience.scuflworkers.java.SplitByRegex` |
| Activity/ConfField/script | List split = new ArrayList();if (!string.equals(””)) { String regexString = ”,”; if (regex != void) ... |
| Processor/ProcessorType | Processor |
| Processor/ProcessorName | reformat_list |

However, the objective of these feature sets is to support the clustering of the annotated IO port pair through finding similarities with IO port pairs to be annotated. At this stage of the study we performed a preliminary evaluation of the distribution of the features extracted. We discovered that very few of them were shared between a significant number of port pairs (see Figure 4). In order to increase the number of shared features we generated a set of *derived features* by extracting bags of words from lexical feature values and by performing Named Entity Recognition on the features that constituted textual annotations (labels and comments), when present. Moreover, from the extracted entities we also added the related DBPedia categories and types as additional features. As example, Table 2 shows a sample of the bag of words and entities extracted from the features listed in the previous Table 1.

The generation of derived features increased the number of total features significantly (up to 59217), while making the distribution of features less sparse, as reported in Figure 5.

**Table 2.** Example of derived features (bag of words and DBPedia entities) generated for the IO port pair 1 → 3.

| **Type** | **Value** |
|---|---|
| From/FromPortName-word | string |
| To/ToPortName-word | split |
| From/FromLinkedPortDescription-word | single |
| From/FromLinkedPortDescription-word | possibilities |
| From/FromLinkedPortDescription-word | orb |
| From/FromLinkedPortDescription-word | mass |
| FromToPorts/DbPediaType | wgs84:SpatialThing |
| FromToPorts/DbPediaType | resource:Text_file |
| FromToPorts/DbPediaType | resource:Mass |
| FromToPorts/DbPediaType | Category:State_functions |
| FromToPorts/DbPediaType | Category:Physical_quantities |
| FromToPorts/DbPediaType | Category:Mathematical_notation |

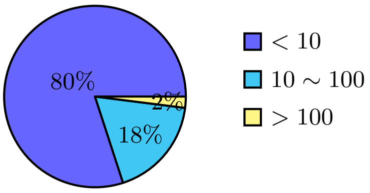

**Fig. 4.** Distribution of features extracted from the workflow descriptions.

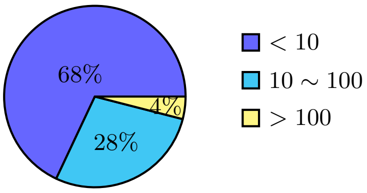

**Fig. 5.** Distribution of features (including derived features).

### 3.3 Retrieval of association rules and generation of recommendations

Generating recommendations usually requires an annotated corpus to be available as training set. While repositories of workflows (especially scientific workflows) exist, they are not annotated with data-to-data relations. In order to overcome this problem we opted for an incremental approach, where the recommendations are produced according to the available annotated items *on demand.* The rules needed are of the following form:

$$(f^1, f^2, ..., f^n) \rightarrow (a^1, a^2, ..., a^n)$$

where $f1, .., fn$ are the features of the IO port pairs and $a1, .., an$ are the data-to-data relations used to annotate them. Our approach relies on extracting association rules from a concept lattice built through FCA incrementally. Such a lattice is built on a formal context of items and attributes. In FCA terms, the items are the IO port pairs and the attributes their features as well as the chosen annotations. Each node of the FCA lattice is a closed concept, mapping a set of items all having a given set of attributes. A FCA concept would then be a collection of IO port pairs all having a given set of features and/or annotations. In a FCA lattice, concepts are ordered from the top concept (supremum), including all items and (usually) no shared features, to the bottom concept (infimum), including all the available features and a (possibly) empty set of items. The lattice is built incrementally using the Godin algorithm [12]. The algorithm (re)constructs the lattice integrating at each iteration a new item - the IO port pair, with its set of attributes (the features and annotations altogether). Association rules are extracted from the FCA lattice, where the key point is the co-occurrence of features $f$ and annotations $a$ in the various FCA concepts.

The following Listing 1.1 gives a sample of an association rule we want to mine from the lattice:

**Listing 1.1.** Example of association rule mined from the FCA lattice.

```
(ProcessorName—word: base,
 FromPortName: base64,
 ActivityName—word: decode,
 ActivityType: http://ns.taverna.org.uk/2010/activity/beanshell,
 ProcessorName—word: array,
 FromPortName—word: base64,
 ToPortName: bytes,
 ActivityName—word: 64,
 ActivityConfField: mavenDependency,
 ActivityName—word: array,
 ActivityConfField: derivedFrom,
 ProcessorName—word: decode,
 ActivityName—word: byte)
   → (dn:hasDerivation, dn:refactoredInto)
```

Several approaches have been studied to generate and rank association rules from a FCA lattice. A common problem in this scenario is the number of rules that can be extracted, and how to reduce them effectively [18]. Indeed, the number of rules can increase significantly with the number of concepts of the FCA lattice. Generating all of them is time consuming as the lattice becomes larger. Precomputing the rules is not a valid solution, as the lattice will change for any new item inserted. In this scenario, we are forced to compute the rules *live* for each new item to be annotated.

The above considerations motivate a set of new requirements for implementing a rule mining algorithm that is effective in this scenario:

1. generate only rules that have annotations in the body
2. generate only rules that are applicable to the candidate item to be annotated
3. only use one rule for each recommendation (head of the rule), to avoid redundancies
4. rank the rules to show the most relevant first

In order to satisfy the requirements above we propose an algorithm to mine association rules *on-demand*, by considering two sets of attributes as constraints for the head and body of the rules.

The algorithm we propose has three inputs: (1) a FCA Lattice; (2) the set of attributes of the item for which we need recommendations (the set of attributes that needs to be in the body of the rules); and (3) the set of attributes we want to be part of the recommendations (the set of attributes that can be in the rule head). Listing 1.2 illustrates the algorithm for extracting rules *on-demand*. Input is a lattice $L$, a set of attributes as possible recommendations (target rule head: $H$) and a set of attributes for which we need recommendations (target rule body: $B$). The algorithm assumes the two sets to be disjoint. The algorithm traverses the lattice starting from the bottom, adding the *infimum* to a FIFO queue - lines 3-5. For each concept in the queue, first assess whether its attributes contains items from both the target head and body. If it doesn't, the concept (and related paths in the lattice) can be skipped - lines 7-11. Otherwise, the parent concepts are added to the queue, and the concept considered to rule extraction - line 13. The non empty intersections of attributes with the target head and body form a candidate rule $b \rightarrow h$.

**Listing 1.2.** Algorithm to mine association rules from a lattice *on demand*:

```
 1  // L: the lattice; H: attributes in the rule head; B: attributes in the rule body
 2  mineRules(L,H,B):
 3    C ← [] // an empty FIFO list of concepts
 4    R ← [] // an empty set of Rules (indexed by their head).
 5    add(inf(L), C) // add the infimum of L to C
 6    while !empty(C):
 7       c ← first(C) // remove one concept from the top of the queue
 8       h=retain(attributes(c),H) // attributes in c in the head of rule
 9       if empty(h): continue // move to another concept
10       b ← retain(attributes(c),B) // attributes in c allowed in the body of the rule
11       if empty(b): continue // move to another concept
12       // Add the concept parents to the queue.
13       addAll(parents(L,c),C)
14       // Examine b → h measures (s: support, k: confidence, r: relevance)
15       // support (s): items satisfying the rule divided by all items
16       s ← count(objects(c)) / count(objects(supremum(L)))
17       if s = 0: continue // A supremum rule includes this one
18       // confidence (k): support divided by the items only satisfying b
19       I ← [] // items only satisfying the body
20       for p in parents(c):
21        if (attributes(p) ∩ h) = ∅:
22          if attributes(p) = b: add(objects(p), I)
23        end
24       end
25       if count(I) = 0: k ← 1
26       else:
27         k ← count(objects(c)) / count(I)
28       end
29       // relevance (r): intersection of B with b, divided by B
30       r ← count(B ∩ b) / count(B)
31       // check this rule is the best so far with this head
32       if hasRuleWithHead(R,h):
33          rule ← getRuleWithHead(R,h)
34          if relevance(rule) > r: continue
35          if relevance(rule) = r:
36              if confidence(rule) > k: continue
37              if confidence(rule) = k:
38                  if support(rule) >= s: continue
39              end
40          end
41       end
42       rule ← (h,b,s,k,r) // the new rule, or the best so far for head
43       add(rule, R)
44    end
45  return R
```

The association rule derived is scored by support ($s$), confidence ($k$) and a third measure inspired from information retrieval and called *relevance* ($r$) - lines 15-30. The definitions of these measures, considering a rule $b \rightarrow h$, is as follows:

- Support $s$ ($b \rightarrow h$): the ratio of items satisfying $b \cup h$ to all the items in the lattice - line 16;

- Confidence $k$ $(b \to h)$: the ratio of items satisfying $b \cup h$ to the items satisfying $b$ - lines 19-28;
- Relevance $r$ $(b \to h)$: the degree of overlap between the body of the rule $b$ and the set of features of the candidate item $B$. It is calculated as the size of the body divided by the size of the intersection between the body of the rule and the features of the candidate item - line 30.

Only the rule with best score for a given head is kept in the list of rules - lines 31-43. Our ranking algorithm will privilege relevance over confidence and support, in order to boost the rules (recommendations) that are more likely to be relevant for the candidate item.

Since this is an iterative process, at the very beginning there will be no recommendation. New annotations will feed the reference corpus (the FCA lattice) and the system will start to generate association rules. Our hypothesis is that the quality of the rules and therefore their usefulness in supporting annotations, increase with the size of the annotated items (this will be part of the evaluation in Section 5).

## 4 Implementation of the approach

The approach described in the Section 3 has been implemented in the Dinowolf (Datanode in workflows) tool[15] based on the SCUFL2 worfklow specification[16] and the taxonomy of data-to-data relations represented by the Datanode ontology. While Dinowolf has been implemented leveraging the Apache Taverna[17] library, it can work with any input following the SCUFL2 specification. When a workflow is loaded, the system performs a preliminary operation to extract the IO port pairs and to precompute the related set of features following the methods described in Sections 3.1 and 3.2. In order to expand the feature set with derived features - bag of words and entities from DBPedia - the system relies on Apache Lucene[18] for sentence tokenization (considering english stopwords), DBPedia Spotlight[19] for named entity recognition, and the DBPedia[20] SPARQL endpoint for feature expansion with categories and entity types. The tool includes three views: 1) a Workflows view, listing the workflows to be annotated; 2) a Workflow details view, including basic information and a link to the external documentation at My Experiments; and a 3) Annotation view, focused on providing details of the features of the IO port pair to annotate. The task presented to the users is the following:

1. Choose an item from the list of available workflows;

[15] Dinowolf: `http://github.com/enridaga/dinowolf`.

[16] SCUFL2 Specification: `https://taverna.incubator.apache.org/documentation/scufl2/`.

[17] Apache Taverna: `https://taverna.incubator.apache.org/`.

[18] Apache Lucene: `https://lucene.apache.org/core/`

[19] DBPedia Spotlight: `http://spotlight.dbpedia.org/`.

[20] DBPedia: `http://dbpedia.org/`.

2. Select an IO port pair to access the Annotation view;
3. The annotation view shows the features associated with the selected IO port pair alongside a list of data node relationships exploiting a set of rules extracted from the FCA lattice as recommendations, and the full Datanode hierarchy as last option;
4. The user can select one or more relations by picking from the recommended ones or by exploring the full hierarchy. Recommended relations, ranked following the approach described in Section 3.3, are offered with the possibility to expand the related branch and select one of the possible subrelations as well;
5. Alternatively, the user can skip the item, if the IO port pair does not include two data objects (it is the case of a configuration parameter set as input for the processor);
6. Finally, the user can postpone the task if she feels unsure about what to choose and wants first explore other IO port pairs of the same workflow;
7. The user iteratively annotate all the port pairs of a workflow. At each iteration, the system makes use of the previous annotations to recommend the possible relations for the next selected IO port pair.

This system has been used to perform the user based experiments that constitute the source of our evaluation.

## 5 Experimental evaluation

Our main hypothesis is that the approach presented can boost the task of annotating workflows as data-to-data annotated graphs. In particular, we want to demonstrate that the quality of the recommendations improves while the annotated cases grow in number. In order to evaluate our approach we performed a user based evaluation. We loaded twenty workflows from "My Experiments"[21] in Dinowolf and asked six users to annotate the resulting 260 IO port pairs. The users, all members of the research team of the authors, have skills that we consider similar to the ones of a data manager, for example in the context of a large data processing infrastructure like the one of [4]. In this experiment, users were asked to annotate each one of the IO port pairs with a semantic relation from a fixed vocabulary (the Datanode ontology), by exploiting the workflow documentation, the associated feature set and the recommendations provided. The workflows were selected randomly and were the same for all the participants, who were requested also (a) to follow the exact order proposed by the tool, (b) to complete all portpairs of a workflow before moving to the next; (c) to only perform an action when confident of the decision, otherwise to postpone the choice (using the "Later" action); (d) to select the most specific relation available - for example, to privilege *processedInto* over *hasDerivation*, when possible. Each user worked on an independent instance of the tool (and hence lattice) and performed the annotations without interacting with other participants. During the experiment the system monitored a set of measures:

[21] My Experiments: `http://www.myexperiments.org`.

- the time required to annotate an IO port pair;
- how many annotations were selected from recommendations;
- the average rank of the recommendations selected, calculated as a percentage of the overall size of the recommendation list; and
- the average of the relevance score of the recommendations selected.

Figures 6-9 illustrate the results of our experiments with respect of the above measures. In all diagrams, the horizontal axis represents the actions performed in chronological order, placing on the left the initial phase of the experiment going towards the right until all 260 IO port pairs were annotated. The diagrams ignore the actions marked as ”Later”, resulting on few jumps in users' lines, as we represented in order all actions including at least one annotation from at least a single user. Figure 6 shows the evolution of the time spent by each user on a given annotation page of the tool before a decision was made. The diagram represents the time (vertical axis) in logarithmic scale, showing how, as more annotations are made and therefore more recommendations are generated, the effort (time) required to perform a decision is reduced. Figure 7 illustrates the progress of the ratio of annotations selected from recommendations. This includes cases where a subrelation of a recommended relation has been selected by the user. While it shows how recommendations have an impact from the very beginning of the activity, it confirms our hypothesis that the cold-start problem is tackled through our incremental approach. Figure 8 depicts the average rank of selected recommendations. The vertical axis represents the score placing at the top the first position. This confirms our hypothesis that the quality of recommendations increases, stabilizing within the upper region after a critical mass of annotated items is produced, reflecting the same behavior observed in Figure 7. Finally, we illustrate in Figure 9 how the average relevance score of picked recommendations changes in time. The relevance score, computed as the portion of features matching a given recommendation that overlaps with the features of the item to be annotated, increases partly because the rules become more abstract (contain less features), partly reflecting the behavior of the ranking algorithm and matching the result of Figure 8.

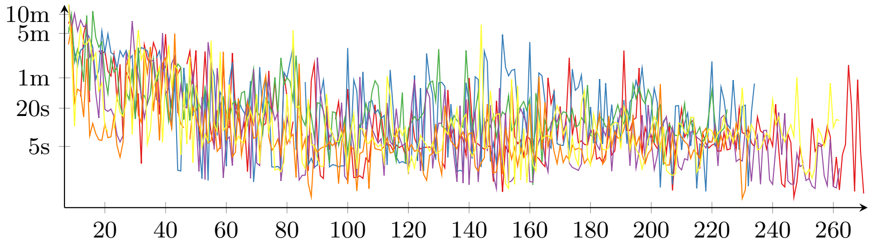

**Fig. 6.** Evolution of the time spent by each user on a given annotation page of the tool before a decision was made.

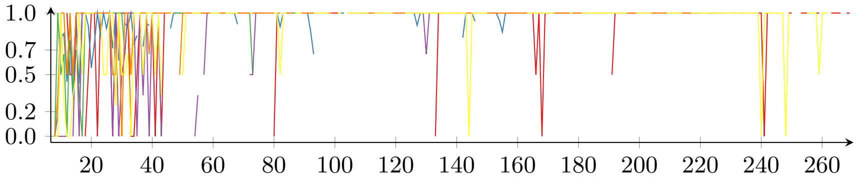

**Fig. 7.** Progress of the ratio of annotations selected from recommendations.

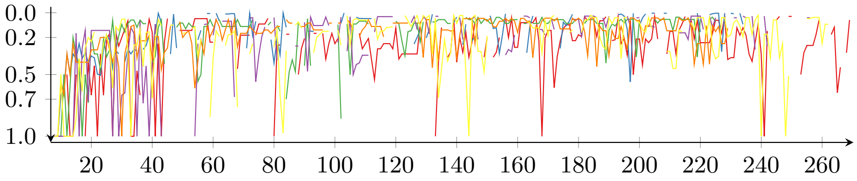

**Fig. 8.** Average rank of selected recommendations. The vertical axis represents the score placing at the top the first position.

## 6 Conclusions

In this article we proposed a novel approach to support the semantic annotation of workflows with data centric relations. We showed through applying this approach on a set of workflows from the My Experiments repository that it can effectively reduce the effort required to achieve this task for data managers and workflow publishers. We plan to integrate the presented approach with the methodology described in [4] in order to support Data Hub managers in the annotation of the data manipulation processes required to compute the propagation of policies associated with the data involved. We have enough confidence to believe that the characteristics of scientific workflows as data intensive workflows [16] are equivalent, because they can be reduced to data centric representations, as demonstrated in Section 3.1.

The quality and consistency of the resulting annotations are not the subject of the present study, and we did not discussed the interpretation of the Datanode relations with the participants of our experiment. For this reason each user operated on a separate instance of the tool, to reduce the possibility that inconsistent usage of relations would negatively impact the quality of the association rules generated. However, we received feedback that encourages to better document the Datanode ontology, for example providing cases of the possible uses and misuses of each relation.

In this work we only focused on the relations between input and output within workflow processors. It is possible to extend this approach to also cover relations between data items with other directions (input to input, output to input, etc...).

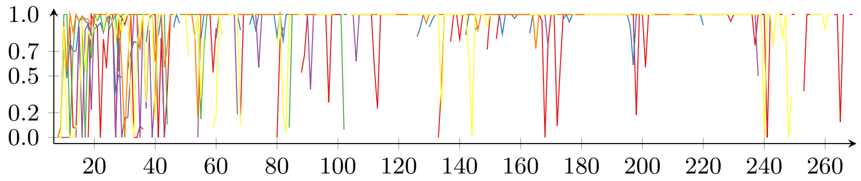

**Fig. 9.** Progress of the average relevance score of picked recommendations.

The FCA component of the Dinowolf Tool is based on an incremental lattice construction algorithm. We plan to integrate a lattice *update* algorithm in order to support modifications to the annotations.

However, the incremental learning of association rules approach presented in this paper is independent from both the features of the item to annotate and the nature of the annotations. This opens the hypothesis that it could be effectively reused in other scenarios.

## References

1. Alper, P., Belhajjame, K., Goble, C.A., Karagoz, P.: Labelflow: Exploiting workflow provenance to surface scientific data provenance. In: International Provenance and Annotation Workshop. pp. 84–96. Springer (2014)
2. Belhajjame, K., Corcho, O., Garijo, D., Zhao, J., Missier, P., Newman, D., Bechhofer, S., Garc a Cuesta, E., Soiland-Reyes, S., Verdes-Montenegro, L., et al.: Workflow-centric research objects: First class citizens in scholarly discourse. In: Proceedings of Workshop on the Semantic Publishing,(SePublica 2012) 9 th Extended Semantic Web Conference Hersonissos, Crete, Greece, May 28, 2012 (2012)
3. Belhajjame, K., Zhao, J., Garijo, D., Garrido, A., Soiland-Reyes, S., Alper, P., Corcho, O.: A workflow prov-corpus based on taverna and wings. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. pp. 331–332. ACM (2013)
4. Daga, E., d'Aquin, M., Adamou, A., Motta, E.: Addressing exploitability of Smart City data. In: Smart Cities Conference (ISC2), 2016 IEEE Second International. IEEE (2016)
5. Daga, E., d'Aquin, M., Gangemi, A., Motta, E.: Describing semantic web applications through relations between data nodes. Tech. Rep. kmi-14-05, Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes (2014), `http://kmi.open.ac.uk/publications/techreport/kmi-14-05`
6. Daga, E., d'Aquin, M., Gangemi, A., Motta, E.: Propagation of policies in rich data flows. In: Proceedings of the 8th International Conference on Knowledge Capture. pp. 5:1–5:8. K-CAP 2015, ACM, New York, NY, USA (2015), `http://doi.acm.org/10.1145/2815833.2815839`
7. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Semantically-aided business process modeling. In: International Semantic Web Conference. pp. 114–129. Springer (2009)
8. Ferreira, D.R., Alves, S., Thom, L.H.: Ontology-based discovery of workflow activity patterns. In: International Conference on Business Process Management. pp. 314–325. Springer (2011)

9. Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: A pattern-based ontology for describing publishing workflows. In: Proceedings of the 5th International Conference on Ontology and Semantic Web Patterns - Volume 1302. pp. 2–13. WOP'14, CEUR-WS.org, Aachen, Germany, Germany (2014), http://dl.acm.org/citation.cfm?id=2878937.2878939
10. Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., Goble, C.: Common motifs in scientific workflows: An empirical analysis. Future Generation Computer Systems 36, 338–351 (2014)
11. Garijo, D., Gil, Y.: A new approach for publishing workflows: Abstractions, standards, and linked data. In: Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science. pp. 47–56. WORKS '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2110497.2110504
12. Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on galois (concept) lattices. Computational intelligence 11(2), 246–267 (1995)
13. Gómez-Pérez, J.M., Corcho, O.: Problem-solving methods for understanding process executions. Computing in Science & Engineering 10(3), 47–52 (2008)
14. Hettne, K., Soiland-Reyes, S., Klyne, G., Belhajjame, K., Gamble, M., Bechhofer, S., Roos, M., Corcho, O.: Workflow forever: Semantic web semantic models and tools for preserving and digitally publishing computational experiments. In: Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences. pp. 36–37. SWAT4LS '11, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2166896.2166909
15. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence 14(2-3), 189–216 (2002)
16. Liu, J., Pacitti, E., Valduriez, P., Mattoso, M.: A survey of data-intensive scientific workflow management. Journal of Grid Computing 13(4), 457–493 (2015)
17. Palma, R., Corcho, O., Hotubowicz, P., Pérez, S., Page, K., Mazurek, C.: Digital libraries for the preservation of research methods and associated artifacts. In: Proceedings of the 1st International Workshop on Digital Preservation of Research Methods and Artefacts. pp. 8–15. DPRMA '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2499583.2499589
18. Poelmans, J., Elzinga, P., Viaene, S., Dedene, G.: Formal concept analysis in knowledge discovery: a survey. In: International Conference on Conceptual Structures. pp. 139–153. Springer (2010)
19. Poelmans, J., Kuznetsov, S.O., Ignatov, D.I., Dedene, G.: Formal concept analysis in knowledge processing: A survey on models and techniques. Expert systems with applications 40(16), 6601–6623 (2013)
20. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proceedings of the 3rd International Workshop on Semantic Business Process Management (SBPM08), CEUR-WS Proceedings. vol. 472 (2008)
21. Wille, R.: Formal concept analysis as mathematical theory of concepts and concept hierarchies. In: Formal Concept Analysis, pp. 1–33. Springer (2005)
22. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., et al.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic acids research p. gkt328 (2013)