Exploring Repositories of Scientific Workflows

Julia Stoyanovich*
University of Pennsylvania
Philadelphia, PA, USA
jstoy@cis.upenn.edu

Ben Taskar
University of Pennsylvania
Philadelphia, PA, USA
taskar@cis.upenn.edu

Susan Davidson†
University of Pennsylvania
Philadelphia, PA, USA
susan@cis.upenn.edu

ABSTRACT

Scientific workflows are gaining popularity, and repositories of workflows are starting to emerge. In this paper we present some initial experiences of information discovery in repositories of scientific workflows. In the first part of the paper we consider a collection of *VisTrails* workflows, and explore how this collection may be summarized when workflow modules are used as features. We present a hierarchical browsable view of the repository in which categories are derived using frequent itemset mining or latent Dirichlet allocation. We demonstrate that both approaches may be used for effective data exploration. In the second part of the paper we focus on a collection of *Taverna* workflows from *myExperiment.org*, and consider how these workflows may be browsed using *modules* and *tags* as features. Finally, we outline some interesting challenges and describe conditions under which these techniques work well for repositories of scientific workflows, and conditions under which additional work is needed for effective data exploration.

1. INTRODUCTION

Workflows have been gaining popularity in life sciences, where users deal with large amounts of data and perform sophisticated *in silico* experiments. Workflow management systems can help automate repetitive tasks and record the provenance of a data product, making the experiment reproducible. A workflow is typically represented by a graph with labeled nodes and typed, labeled edges. Nodes correspond to functional modules, and edges represent the data flow between modules. In addition to the graph structure, workflows may be characterized by a variety of other attributes, such as a text description of the workflow as a whole, a description of each module, a hierarchical relationship between the modules representing sub-workflows, tags assigned to the workflow by its creator or by other users, types of input and output data that the workflow generates, etc.

A scientist who wishes to use a workflow development platform may opt for one from among several popular solutions such as *Kepler* [5], *Taverna* [16], *VisTrails* [6], or *geWorkbench* [15], to name just a few. A scientist may also share his workflows by posting them to *myExperiment.org* [19], a collaborative platform for the exchange of scientific workflows and experimental plans, or to the Kepler Component Repository (`library.kepler-project.org/kepler/`).

Consider Joe, a novice user of a popular workflow management system *Taverna* [16]. Joe wishes to develop a workflow that takes as input gene identifers and looks up *KEGG* [12] pathways relevant to the query genes. The workflow should then identify other genes that are relevant to the retrieved pathways, and query *OMIM* [14] for records that describe heritable disorders to which any of the genes are linked. Joe would like to see examples of workflows that implement similar functionality, demonstrating which processing modules may need to be invoked, and how to transform the data at intermediate processing steps so that the output of a *KEGG* lookup may be used as a search query in *OMIM*.

Joe visits *myExperiment.org* and looks for relevant workflows by issuing a search query, e.g. "OMIM KEGG". As a collaborative platform, *myExperiment.org* implements a variety of social web features. For example, workflows are tagged by its creator and by other users of the system, allowing for more effective searching of the corpus. Joe's query may therefore return workflows in which "OMIM" or "KEGG" are part of the *description*, correspond to *module names*, or occur among the *tags* of the workflow. Which of these features are used to answer the query depends on the implementation of the search functionality. *The focus of our work is to study the effect of using tags and module names as features.*

Suppose that Joe was able to identify a set of interesting workflows as a result of his search. He would, however, like to expand this set with other workflows that are potentially interesting even if they did not precisely match his query. After all, asking a high-recall query is a challenging task, since it requires that the user is both well-aware of his information need and able to formulate it precisely in terms of the available features. These expectations are often unrealistic, particularly for novice users like Joe. Indeed, in scientific research, *knowing what question to ask and how to ask it amounts to solving a significant part of the problem.* There is therefore a need for intuitive data exploration tools

*This material is based in part upon work supported by the National Science Foundation grant 0937060 to the Computing Research Association for the CIFellows Project.
†This material is based in part upon work supported by the National Science Foundation grant IIS 0803524.

WANDS Indianapolis, IN, USA, June 6th, 2010

that facilitate information discovery.

A particular family of data exploration tools summarizes collections one feature at a time. A *tag cloud* presents a set of all, or of the most frequent, tags used in the collection, with font size corresponding to the popularity of a tag. Tag clouds are currently used by a variety of social tagging sites, including *myExperiment.org*. A similar presentation may, of course, be used for discrete features other than tags. For example, a *module cloud* may be generated for a scientific workflow collection, in which the most frequently used modules would be the most prominent.

A significant drawback of a cloud is that using all available features may overwhelm the user, preventing him from finding the features of interest. There are often thousands of features, such as tags or module names, in use in a particular community – too many to present in an unstructured manner. On the other hand, if only a handful of most popular features are selected, they may not be relevant to a specific information need.

Another method that summarizes collections one feature at a time is *faceted browsing*, which is based on a hierarchically organized collection of mutually exclusive, collectively exhaustive properties called *facets* [23]. Faceted hierarchies are used extensively to organize product catalogues (e.g., `Amazon.com`), scientific articles [11], archaeological finds [18], etc. Note that, while facets represent orthogonal properties of items in a collection (e.g., color and make of a car), an item may belong to one or more facets (e.g., a particular car may be both *red* and *a Ferrari*). In contrast to *monolithic hierarchies* in which categories are disjoint, the same item may be reachable through several paths in a faceted hierarchy.

A faceted representation has an obvious advantage over a feature cloud. It alleviates the *too many features* problem by organizing features into a hierarchy, allowing navigation at multiple levels of granularity. However, to enable faceted browsing of workflow collections, one must first ensure that a suitable hierarchy over the features is available, or can be derived. For features of interest to us, namely tags and workflow modules, no pre-defined hierarchies exist to the best of our knowledge.

Recall that faceted browsing is still essentially a *feature-at-a-time* method. This aspect of the model may limit the effectiveness of data exploration because it does not explicitly capture the legal, or common, combinations of features. For example, while make and color of a car are, in general, orthogonal properties, certain combinations of values are far more likely than others, for certain cars (e.g., Ferraris are often red). This analogy also applies to scientific workflows, where a *myExperiment.org* workflow tagged with *KEGG* will also likely be tagged with *pathways*. Similarly, a *VisTrails* workflow that computes a histogram is likely to subsequently display a histogram on the screen. Meaningful combinations of features, which we term *categories*, can greatly assist the user in identifying interesting items, and we term this type of data exploration *category-at-a-time*.

An assignment of features to categories may be known *a priori*, or it may be derived from the data. In our application scenario, categories are not known in advance. *An important question is then: how can appropriate categories of features be derived and organized hierarchically, to enable browsing?* This paper takes a first step in this direction for collections of scientific workflows. We consider two collections: *Taverna* workflows that are publicly available on *myExperiment.org*, and *VisTrails* workflows created by students of a scientific visualization course at the University of Utah. Our goals are to use the data to derive labeled categories (combinations of features), to be used for data exploration. As with faceted classification, categories describe orthogonal properties, and items may belong to several categories. Ideally we would also like to provide a hierarchical view of the categories. We will use two methods, frequent itemset mining and latent Dirichlet allocation, to advance towards these goals.

We also note that clustering is a well-known data exploration paradigm. Many clustering algorithms exist, and the *category-at-a-time* approaches that we use may also be considered a type of clustering. In [21] several distance-based clustering approaches are explored for the collection of *VisTrails* workflows that we also use in our experiments. Distance-based clustering may be termed *all features at-a-time* in our terminology, because all features together make up a distance between two items. Distance-based clustering differs from our approach in that the clusters are not easily labeled. Also, each cluster represents a set of items that are similar along all dimensions, and does not discriminate between aspects of functionality, and aspects of similarity. Finally, each item is assigned to exactly one cluster.

This paper makes the following contributions. First, we propose to use frequent itemset mining and topic modeling, a state of the art data exploration technique, for exploring repositories of scientific workflows, and adapt these techniques to our setting. Second, we present results of an evaluation of appropriateness of these techniques on two real repositories of scientific workflows. Third, we discuss the conditions under which these data exploration techniques work well, and conditions for which additional techniques may need to be developed.

The remainder of this paper is organized as follows. We describe the datasets in Section 2, and present our proposed technical approach in Section 3. Section 4 details our experimental evaluation. We discussion our findings and intuitions in Section 5 and conclude in Section 7.

2. DESCRIPTION OF THE DATASETS

We now describe two datasets that motivate our work. The first is a thematically focused collection of *VisTrails* workflows that contains workflow specifications with a limited amount of meta-data. The workflows in this collection deal primarily with information visualization tasks. We will explore the use of *modules* as features, and will attempt to derive categories of modules for exploring this dataset.

The second is a collection of *Taverna* workflows downloaded from *myExperiment.org* on February 16, 2010, consisting of workflow specifications and rich tagging information. This dataset is smaller in size than the first, and covers a broader range of functionality. Here, we will explore the utility of using both *tags* and *modules* as features.

2.1 VisTrails

VisTrails is an open-source scientific workflow and provenance management system developed at the University of Utah [6]. *VisTrails* represents a workflow with a graph, where labeled nodes correspond to *processing modules*, and typed edges stand for *dataflow connections* between pairs of modules. A distinguishing feature of *VisTrails* is the detailed recording of the provenance of data products, workflow spec-

ifications, and workflow executions [6].

VisTrails has been used as an instructional platform for a scientific visualization (SciVis) course at the University of Utah. The *VisTrails* team generously shared with us the workflows that were created by the students in this course. Our data analysis in this section, and the evaluation of our proposed methods in Section 4.1, is based on the Fall 2007 SciVis dataset, the same dataset as was used in [21].

Over the course of the semester the students were required to submit 5 homework assignments and a final project, with each assignment consisting of several problems. The problems were manually classified by the course staff as relating to one of thirteen *classes* that describe the type of a problem the students were solving [21]. Students submitted both the final solution for each problem, and the complete workflow provenance, i.e., workflows that were created at intermediate development steps. As was done in [21], we identified the workflows that corresponded to the final solution for each problem, and we only use these workflows in our evaluation. Whether or not a workflow corresponds to a final solution is determined by looking at a label assigned to the workflow by the student, e.g., "Problem 1a".

Our final dataset consists of 2075 distinct workflows. This number is higher than 1730, as reported in [21], since we were unable to match their labeling precisely due to formatting inconsistencies. However, the distribution of the number of workflows per class label stayed approximately the same. The workflows in our dataset used 129 distinct processing modules. Workflows sizes, in terms of the total number of modules, ranged from 1 to 128, with 15.5 modules on average, and a median of 12 modules per workflow.

Figure 1 summarizes module frequencies in the collection. Note that the frequencies are cumulative, e.g., 62 modules were used in at most 20 distinct workflows, while 77 modules were used in at most 50 distinct workflows. Modules that appear in fewer than 20 workflows (1% of the dataset) are unlikely to be useful as part of a category, because the module by itself does not index a significant portion of the dataset. Likewise, modules that are too common, e.g., those that appear in more than 500 workflows (25% of the dataset) are unlikely to be useful because they do not discriminate well between different types of workflows. Based on the data in Figure 1, where a significant number of modules appears in between 1% and 25% of the dataset, we conclude that using modules to organize this workflow collection into categories is promising.

2.2 myExperiment

The second dataset with which we work is a collection of *Taverna* workflows that have been posted to *myExperiment.org*, together with some associated meta-data. We focus our attention on the *Taverna* sub-set of *myExperiment.org* because these workflows constitute close to 90% of the repository at the time of this writing. We downloaded 663 distinct *Taverna* workflows, a total of 861 workflows if versions are considered.

Taverna is a popular open-source workflow management system created by the ^{my}Grid project. A *Taverna* workflow consists of processor nodes and two types of edges: *link*, representing data flow; and *coordination*, representing a coordination constraint between processors. Workflow-level inputs and outputs are also nodes in the workflow graph, but will be disregarded in our analysis. (See [16] for a detailed description of the *Taverna* data model.)

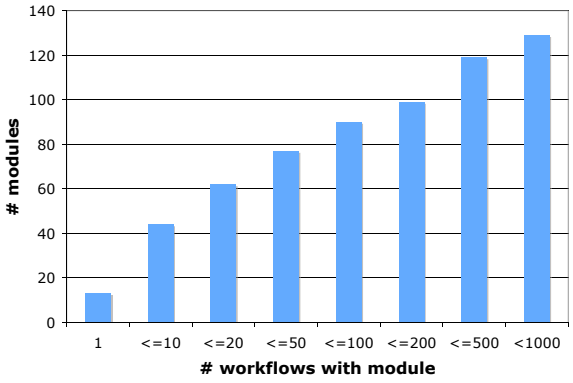

Figure 1: VisTrails: module frequency distribution in the *SciViz 2007* dataset.

Processor nodes in *Taverna* roughly correspond to modules in *VisTrails* in that they define a single atomic processing operation within a workflow. Common types of processors are [1]:

- *stringconstant* typically represents a parameter or a configuration option. We do not use sting constants as features because having a variable by the same name, e.g., "foo", in two workflows, does not necessarily mean that the two workflows are in any way related.
- *arbitrarywsdl* defines access to a standard SOAP-based web service, referenced by the URL to a WSDL document and the operation name within that document to access. We use the combination of a URL and operation name as a feature, e.g., *http://soap.genome.jp/KEGG.wsdl get_motifs_by_gene*.
- *soaplabwsdl* defines an operation using a tool provided by Martin Senger's Soaplab software. We use the full operation URL as a feature, e.g., *http://phoebus.cs.man.ac.uk:1977/axis/services/qtl_analysis.getcurrentdatabase*
- *biomobywsdl* defines an operation using a biomoby service. We use the moby endpoint URL together with a service name as a feature, e.g.,*http://moby.ucalgary.ca/moby/MOBY-Central.pl getKeggPathwayAsGif*.
- *local* defines an operation running in the user's address space based on a local Java class. Local processors often refer to classes and methods that came with the *Taverna* workbench, and so the same method is often invoked by more than one workflow. We use the combination of class name and method name as a feature, e.g., *org.embl.ebi.escience.scuflworkers.java.SplitByRegex*.
- *workflow* Defines an operation using a nested workflow. We use the location of the workflow as a feature, e.g., *http://www.myexperiment.org/workflows/167/download?version=1*.

[1]Descriptions based on `http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html\#processor`, downloaded on 02/26/2010.

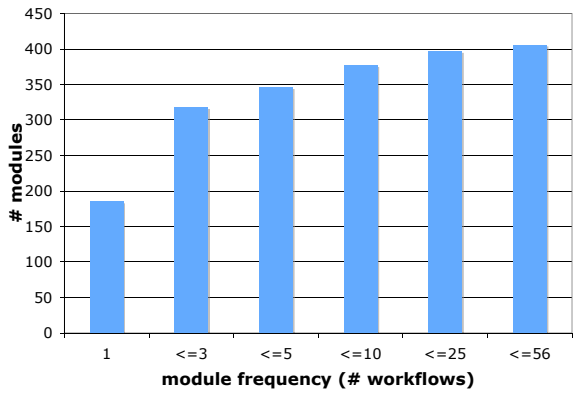

Figure 2: Module frequency for *Taverna* workflows in *myExperiment.org*.

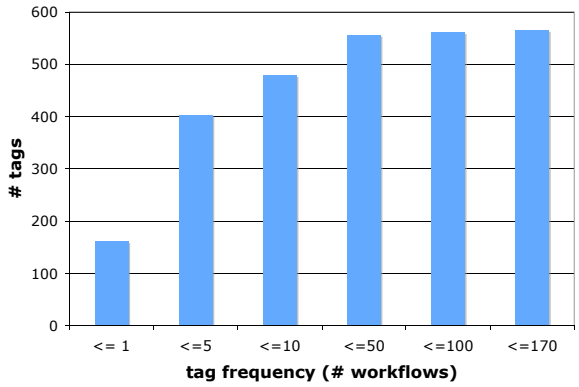

Figure 3: Tag frequency for *Taverna* workflows in *myExperiment.org*.

For consistency of terminology, we will refer to processor-derived features as *modules*. The dataset includes 406 distinct modules, used in 861 workflows. Figure 2 presents module frequencies, by counting the total number of workflows in which a module appears. We notice that the single most frequent module appears in 56 workflows, or only 6.5% of the dataset. Most modules appear in no more than 3 workflows. These statistics are discouraging, because even individual modules do not index a significant portion of the dataset, and so it is unlikely that interesting combinations of modules can be found. Nonetheless, we will attempt to use modules, either on their own or in combination with tags, for browsing.

myExperiment.org also supports tagging, and we will use tags that are associated with *Taverna* workflows as another feature. A total of 565 distinct tags have been assigned to the workflows in our dataset. Workflows are tagged with at least 0 and at most 28 tags. 747 of a total of 861 *Taverna* workflows are tagged, yielding an average of 5 tags and a median of 4 tags per workflow.

Figure 3 presents tag frequencies, as the total number of workflows that have been assigned a particular tag. We see that 161 tags are used in only one workflow, while 403 tags are used in at most 5 workflows; these tags are unlikely to be useful for deriving categories. However, many tags (162 in all) appear in more than 5 workflows, and should therefore be useful for browsing.

3. GENERATING CATEGORIES

In this section we describe two techniques, frequent itemset mining and topic modeling, that we will use to identify feature categories appropriate for browsing. The categories will be labeled, that is, they will be described by the features that generate them. In both cases, categories will index possibly overlapping portions of the dataset.

3.1 Frequent Itemset Mining

The first categorization method we employ is frequent itemset mining. The idea is to find sets of two or more features (in our case, modules or tags), that *frequently co-occur* in workflows. For example, *KEGG* may often tag workflows that are also tagged with *pathway*. Intuitively, frequently co-occurring features represent a conceptual unit, which naturally maps to a browsing category. An itemset is considered frequent if it passes a given *support* threshold, i.e. the percentage of the over-all dataset in which the features co-occur. Suppose that we are given a dataset consisting of 100 workflows, 33 of which are tagged with *KEGG*, 22 are tagged with *OMIM*, and 12 are tagged with both *KEGG* and *OMIM*. The itemset $\{KEGG, OMIM\}$ passes a support threshold of 10%, but it does not pass a threshold of 20%. Both single-feature itemsets pass a 20% support threshold.

We implemented the classic Apriori [1] algorithm to mine frequent itemsets of features. Apriori takes *support* as input, and proceeds in a bottom-up manner. During the first round, the algorithm identifies all frequent itemsets of size 1 (a single feature), by simply counting feature frequencies and pruning out features that do not pass the support threshold. During the k^{th} round the algorithm will attempt to generate itemsets of size k by combining *compatible* frequent itemsets of size $k-1$, and testing whether the resulting k-itemset passes the support threshold. Two itemsets I_1 and I_2 of size $k-1$ are *compatible* if $k-2$ of their features are the same, and the $k-1^{st}$ feature of I_1 is lexicographically lower than the corresponding feature of I_2. For example, consider itemsets $I_1 : \{KEGG, pathway\}$ $I_2 : \{KEGG, OMIM\}$ and $I_3 : \{BLAST, alignment\}$. I_1 and I_2 are compatible, while I_2 and I_3 are not. Itemsets I_1 and I_2 will be combined, and the number of workflows that are tagged with *KEGG*, *pathway* and *OMIM* will be counted. If the itemset of size 3 passes the support threshold, this itemset will be considered frequent. The algorithm terminates when no frequent itemsets are identified during a particular round.

Frequent itemsets naturally form a partial order that may be used to organize the categories into a hierarchy. For example, each item that is indexed by the 3-itemset $\{KEGG, pathway, OMIM\}$ is also indexed by $\{KEGG, pathway\}$, $\{KEGG, OMIM\}$, and $\{pathway, OMIM\}$.

3.2 Latent Dirichlet Allocation

As we discussed in the Introduction, our goal is to allow for the browsing of workflow repositories using categories – combinations of one or several features that together represent a conceptual unit of functionality. These conceptual units are the goals that the user had in mind when he was building the workflow, e.g., sequence alignment, visualization of an iso-surface, building the speciation taxonomy of several related

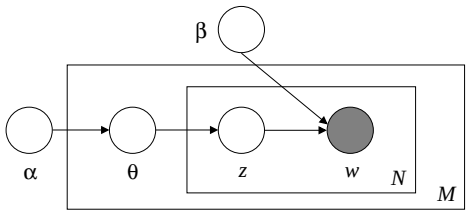

Figure 4: Plate representation of the LDA model.

species, etc. Categories form a *semantic structure* of a workflow collection, with each particular workflow implementing one or several of the topics. If the precise set of categories in the collection were known, together with a mapping between categories and workflows, the collection could be organized around the categories for browsing. However, it is often unreasonable to expect that this the category composition of a workflows collection be known a priori. Typically, this semantic structure is hidden, or latent, and must be learned.

Topic models are probabilistic models for uncovering the underlying semantic structure of a collection based on hierarchical Bayesian analysis, and have been originally proposed for use in text collections. Topics correspond to what we call categories, and we use *categories* and *topics* interchangeably. In our work we use a particular kind of a topic model, called latent Dirichlet allocation (LDA) [4]. LDA is a generative model that allows sets of observations to be explained by unobserved (latent) random variables. In our case, observations are the presence or absence of a particular feature (tag or module) in the workflow. LDA assumes that the number of topics in a collection is given, and that topics are drawn from a multinomial distribution with a Dirichlet prior. Each topic is assumed to be drawn independently.

Figure 4 represents LDA using plate notation, with N features and M topics. The outer plate represents workflows, while the inner plate represents the repeated choice of topics and features within a workflow. α is the parameter of the uniform Dirichlet prior on the per-workflow topic distributions. LDA assumes the following generative process for each workflow. First, choose $\theta \sim Dirichlet(\alpha)$. Next, for each of the N features in the vocabulary, choose a topic $z_n \sim Multinomial(\theta)$. Finally, choose a feature w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

As is typically done in topic modeling literature, topics are described by their most probable features. In our evaluation we use the implementation of LDA by David Blei, available at `http://www.cs.princeton.edu/~blei/lda-c`.

4. RESULTS

4.1 VisTrails

For all methods, we pre-processed our dataset by removing modules that appear in over 25% of the workflows. This amounted to removing 9 high-frequency modules, leaving 120 distinct modules. This technique, known as *stop-word elimination*, is an accepted way to filter out features that are too common and are therefore expected to have low utility in discriminating between categories.

Recall that the *VisTrails* dataset consists of solutions to course assignments of SciViz, taught at the University of Utah. Individual problems were labeled as belonging to 1 of 13 classes, and while we are not experts on either the VisTrails platform, or on scientific visualization, we can use the labeling to get an intuition about the quality of categories. We do not expect that categories will always precisely map to class labels. Categories represent topics, or themes – conceptual units of functionality that a workflow implements. A workflow may implement multiple topics, and some of the same topics may need to be implemented by workflows in different classes. Therefore, our expectation is that, while some of the topics will map predominantly to a single class, some other topics will contain an interesting mixture of classes.

Starting from the cleaned dataset, we ran latent Dirichlet allocation (LDA) using module names as features. The LDA algorithm takes the number of topics as a parameter. We experimented with 10, 20, and 50 topics, and our intuition was that the 10 topic setting did not capture the richness of the data sufficiently. For example, we were not able to identify one or more topics that focused on *infovis* and *tetravolume*, smaller classes that correspond to fewer than 5% of the dataset. LDA with 20 and 50 topics produced many similar topics, however, some interesting relationships between topics were more apparent for the higher setting, and this is the result we present here.

The outcome of LDA is a probability distribution that relates features and documents to topics. A topic can typically be described by the few features that it generates with highest probability. In fact, a topic always specifies a valid conditional probability distribution, in the sense that the probabilities of individual features given the topic sum to 1. In our *VisTrails* experiment we propose to describe a topic using features that together make up 80% of the topic's probability mass. We refer to these as the topic's *top features*.

A simple way to map a topic to a class label is to consider all workflows that contain the topic's *top features*. For example, if a topic generates module *MplPlot* with probability 0.507 and module *MptFigureCell* with probability 0.463, we may retrieve all workflows that contain both modules, and look at the distribution of class labels among those workflows. Table 1 presents topics that have at least a 95% probability of generating workflows in a single class, along with the corresponding class label. Several other single-class topics were identified, involving the *vector_field*, *plot*, and *volume_rendering* class labels, but we do not list them here due to space constraints.

Figure 5 presents topics that describe concepts common to two or more classes, along with some hierarchical relationships that hold between the topics. These relationships are useful for organizing the topics into a hierarchy for browsing. We do not list all such classes here due to space constraints.

We also ran the Apriori algorithm to mine frequent itemsets of modules from this dataset. We experimented with several settings for the support threshold. For minimum support of 10% the algorithm finds 457 distinct categories, raising the issue of which categories should be selected for presentation. Setting the support threshold to 2% yields 7313 distinct categories.

Many of the categories generated by Apriori index workflows with a single class label. So, 99% of the workflows indexed by itemsets $\{File, MplFigure\}$, $\{File, MplFigure, MplFigureCell\}$, and $\{File, MplFigure, MplFigureCell, MplPlot\}$ were labeled with *plot*. Clearly, these itemsets may be presented hierarchically.

Topic		
feature	**P(feature)**	**Class Label**
vtkGraphToPolyData	0.290	infovis
vtkGraphLayout	0.259	
vtkActor2D	0.138	
vtkDynamic2DLabelMapper	0.126	
vtkInteractionHandler	0.335	volume_rendering
vtkImplicitPlaneWidget	0.333	
vtkPlane	0.317	
vtkVolume	0.267	volume_rendering
vtkVolumeProperty	0.252	
vtkPiecewiseFunction	0.248	
vtkVolumeTextureMapper3D	0.120	
vtkVolumeTextureMapper2D	0.112	
File	0.999	plot
vtkGlyph3D	0.376	vector_field
vtkArrowSource	0.287	
vtkPointSource	0.089	
vtkSphereSource	0.474	vector_field
vtkTransformFilter	0.163	
vtkRungeKutta4	0.126	
vtkStreamTracer	0.108	
vtkStreamTracer	0.272	vector_field
vtkTransform	0.229	
vtkRungeKutta4	0.220	
vtkTransformFilter	0.200	
vtkOutlineFilter	0.459	tetravolume
vtkQuadric	0.183	
vtkClipVolume	0.179	
vtkSampleFunction	0.179	
vtkWarpScalar	0.381	diff_scalar_field
vtkImageGradientMagnitude	0.186	
vtkLODActor	0.145	
vtkLookupTable	0.130	

Table 1: *VisTrails*: LDA topics that map to a single class label.

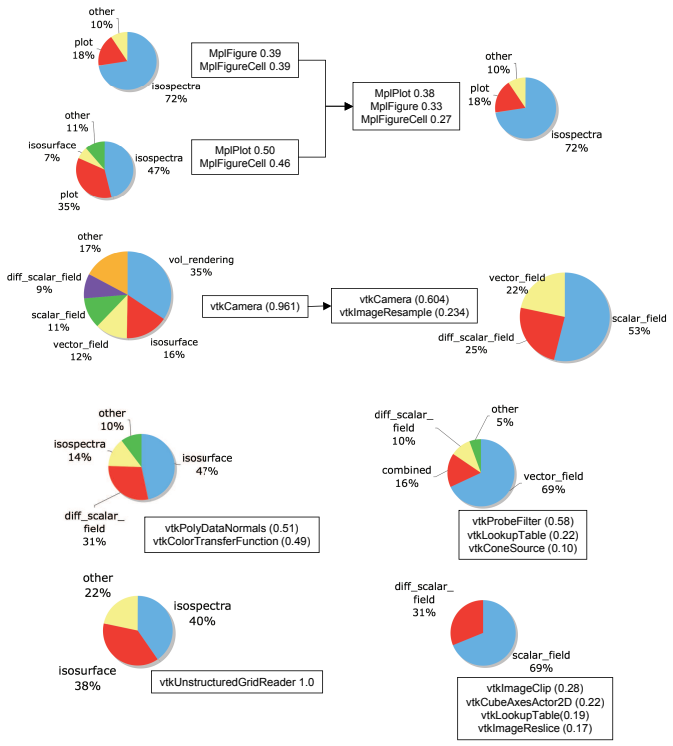

Figure 5: *VisTrails*; LDA topics that map to multiple class labels (best viewed in color).

Itemsets $\{vtkArrowSource, vtkTubeFilter\}$, $\{vtkGlyph3D, vtkRungeKutta4\}$, $\{vtkGlyph3D, vtkSphereSource\}$, $\{vtkRungeKutta4, vtkStreamTracer, vtkTransform, vtkTransformFilter\}$, and many other combinations of these and related modules index workflows with the label *vector_field*, and may also be organized hierarchically using a lattice.

In fact, the majority of itemsets index workflows with a single class label, and that label is either *plot* or *vector_field* workflows, as these are the two largest classes. Unlike LDA, Apriori does not identify as many single-class categories for smaller classes, and it does not find many categories with an interesting mix of class labels. We thus conclude that LDA is better-suited for generating browsing categories for the VisTrails collection.

4.2 myExperiment

To categorize the collection of Taverna workflows downloaded from *myExperiment.org*, we started by using frequent itemset mining to identify combinations of two or more features. A total of 173 categories were mined from the collection of 442 workflows, with support threshold set to 1%. 140 of these consisted exclusively of tags, 40 used only processing modules, and 7 combined tags and modules. Table 2 presents some of the itemsets. We note that all itemsets had fairly low support, meaning that only a handful of workflows would be categorized under each category. The itemset with highest support, *tag:example tag:mygrid*, was relevant to only 6% of the workflows. (Both of these tags are fairly uninformative.) We only list one itemset in Table 2 that uses modules as features, and note that the remaining module-only itemsets were centered around KEGG and pathway analysis.

In our second experiment we applied latent Dirichlet allocation to the collection, using tags as features. Table 3 presents the top-10 tags for 10 topics that were discovered by LDA. We note that most of the topics intuitively correspond to a coherent concept, and we believe that this technique is promising for organizing collections such as *myExperiment.org*.

In our final experiment we applied LDA to the collection, using modules as features. Table 4 presents the top-10 modules for 3 topics. We do not present all 10 topics due to space considerations.

Based on the experimental results in this section we conclude that using tags as features is promising. Using modules as features, either on their own or in combination with tags, yields some interesting categories, but, because a wide variety of different modules is used in this fairly small collection, the utility of modules as features for data exploration is limited. We further discuss this point in Section 5. Finally, based on our results we believe that LDA is better suited than frequent itemset mining for deriving a manageable number of meaningful categories over the *myExperiment.org* collection.

5. DISCUSSION

Summary of results and applicability. We saw in Section 4.1 that frequent itemset mining, and particularly

tag-only
tag:example tag:localworker tag:mygrid
tag:bioinformatics tag:ebi tag:protein
tag:pathway tag:kegg
tag:pathway tag:pathway-driven tag:pathways
tag:kegg tag:pathway-driven tag:phenotype
tag:protein tag:text_mining
tag:bioinformatics tag:ebi tag:protein tag:protein annotation
tag:AIDA tag:BioAID tag:protein tag:text_mining tag:text_mining_network
tag and module
soaplab phoebus.cs.man.ac.uk:1977/axis/services/seq_analysis.parse_ddbj_gene_info tag:benchmarks
wsdl xml.nig.ac.jp/wsdl/Ensembl.wsdl:getGeneInfo tag:benchmarks
wsdl soap.genome.jp/KEGG.wsdl:btit tag:kegg
wsdl soap.genome.jp/KEGG.wsdl get_pathways_by_genes tag:kegg
wsdl ws.adaptivedisclosure.org/axis/services/NERecognizerService?wsdl NErecognize tag:AIDA tag:BioAID
module-only
wsdl:soap.genome.jp/KEGG.wsdl bconv binfo btit get_pathways_by_genes

Table 2: *myExperiment*: frequent itemsets of tags and modules.

topic 1	**topic 2**	**topic 3**	**topic 4**	**topic 5**
BLAST	pathways	ssearch	currency	species
disambiguation	pathway-driven	text	mygrid	bio2rdf
annotation	condition	GO	localworker	ncbi
ensembl	species	abstracts	benchmarks	Kegg pathways
design pattern	genotype	ncbi	newcastle	citation
biomart	protein motif	banff_manifesto	RNA	eb-eye
condition	shim	text mining	floss	ssearch
R	data-driven	beanshell	hcls2009	knowledgescope
rshell	pccompound	remove	utility	sentence breaker
sub-graph	reaction enumeration	mining	communication network	uniprot
topic 6	**topic 7**	**topic 8**	**topic 9**	**topic 10**
image	AIDA	annotation	kalign	bio2rdf
graph	BioAID	barcode	grid service	workflow
e-science	protein	protein	graves	knowledgescope
oxl	tutorial exercise	BLAST	condition	phylogenomics
filter	VL-e	alignment	cagrid	snps
annotation	parser	snps	classification	species
protein	demo	protein annotation	globus	ls-snp
barcode	emboss	taverna	bind	srs
functional workflow	biorange_nl	mesh	affymetirx	search
protein annotation	nbiconworkflows	sequence similarity search	open source software	bfind

Table 3: *myExperiment*: LDA with 10 topics, using tags as features.

topic modeling, may be used to discover browsing categories for the *VisTrails* collection, when modules are used as features. Latent Dirichlet allocation was effective at identifying interesting categories that map to a single class label (see Table 1), as well as categories that point to common functionality across classes (see Figure 5). As we discussed in Section 4.2, LDA generated some interesting topics for the *Taverna* subset of *myExperiment.org*, when tags were used as features. However, using processing modules as features was less effective, both with LDA and with frequent itemset mining. Overall, we feel that results for the *Taverna* subset of *myExperiment.org* were less promising than those for *VisTrails*.

Our intuition is that the *VisTrails* collection is amenable to the proposed techniques because it is fairly focused, covering only a handful of classes / tasks. In contrast, the *Taverna* workflows from *myExperiment.org* cover a wide range of tasks. This, combined with the fact that the *myExperiment.org* collection is fairly small, limits the applicability of our methods. These results support our initial intuitions in Section 2, where we considered feature frequencies in the two collections.

Our conclusion is that the techniques described in this paper will be more effective in repositories that are focused on a small number of tasks. Such repositories may, for example, arise in the context of a research lab, or a class. For these methods to be effective in an open collaborative environment such as *myExperiment.org*, much larger repositories are needed. Nonetheless, we believe that categories such as those derived by LDA with tags as features (see Table 3) will improve the user experience in *myExperiment.org*, potentially encouraging user participation, and supporting growth of the repository.

An interesting follow-up question is how the categories derived by LDA and Apriori should be used in scope of a complete browsing solution for workflow repositories. In particular, the following are some of interesting directions for future work.

Model selection. LDA takes two parameters as input: the number of topics, and the initial value for the parameter α used by the Dirichlet prior. In our evaluation we chose settings for these parameters mostly based on inspect-

topic 1
wsdl:http://soap.genome.jp/KEGG.wsdl:btit
wsdl:http://soap.genome.jp/KEGG.wsdl:bconv
wsdl:http://soap.genome.jp/KEGG.wsdl:get_pathways_by_genes
wsdl:http://phoebus.cs.man.ac.uk:8081/axis/EnsemblListner.jws?wsdl:lister
wsdl:http://soap.genome.jp/KEGG.wsdl:binfo
soaplab:http://phoebus.cs.man.ac.uk:1977/axis/services/qtl_analysis.getcurrentdatabase
wsdl:http://flosseb.floss.syr.edu/taverna/wsdl:MatrixBuilderR
wsdl:http://flosseb.floss.syr.edu/taverna/wsdl:GetPeriods
wsdl:http://flosseb.floss.syr.edu/taverna/wsdl:EventsForProjectsInPeriod
soaplab:http://phoebus.cs.man.ac.uk:1977/axis/services/qtl_analysis.flatten_pathway_files
topic 2
wsdl:http://eutils.ncbi.nlm.nih.gov/entrez/eutils/soap/eutils.wsdl:run_eSearch
wsdl:http://www.chemspider.com/MassSpecAPI.asmx?WSDL:GetExtendedCompoundInfoArray
soaplab:http://phoebus.cs.man.ac.uk:1977/axis/services/linking.srslinks:
wsdl:http://www.chemspider.com/Search.asmx?WSDL:SimpleSearch
wsdl:http://www.chemspider.com/Search.asmx?WSDL:GetCompoundThumbnail
wsdl:http://xml.ddbj.nig.ac.jp/wsdl/Blast.wsdl:searchSimple
soaplab:http://phoebus.cs.man.ac.uk:8081/axis/services/seq_analysis.genscan:
biomoby:http://moby.ucalgary.ca/moby/MOBY-Central.pl:AffyArrayQualityAnalysis_poll
soaplab:http://www.ebi.ac.uk/soaplab/services/edit.skipseq:
soaplab:http://www.ebi.ac.uk/soaplab/services/edit.sizeseq:
topic 3
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSCensor.wsdl:poll
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSFasta.wsdl:poll
wsdl:http://www.ebi.ac.uk/collab/mygrid/service1/goviz/GoViz.jws?wsdl:markTerm
wsdl:http://rguha.ath.cx:8080/cdkws/services/Utility?wsdl:getFragmentWithClosure
wsdl:http://www.ebi.ac.uk/collab/mygrid/service1/goviz/GoViz.jws?wsdl:addTerm
soaplab:http://phoebus.cs.man.ac.uk:1977/axis/services/seq_analysis.blastsimplifier:
wsdl:http://mygrid.ncl.ac.uk/axis/services/SrsEbiQuery?wsdl:queryByXRef
wsdl:http://www.ebi.ac.uk/collab/mygrid/service1/goviz/GoViz.jws?wsdl:getChildren
wsdl:http://www.ebi.ac.uk/collab/mygrid/service1/goviz/GoViz.jws?wsdl:destroySession
wsdl:http://www.ebi.ac.uk/collab/mygrid/service1/goviz/GoViz.jws?wsdl:getDot
topic 4
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSInterProScan.wsdl:poll
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSInterProScan.wsdl:checkStatus
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSInterProScan.wsdl:runInterProScan
wsdl:http://egee1.unice.fr/wsdl/gasw_service.wsdl:GASWexecution
wsdl:http://eutils.ncbi.nlm.nih.gov/entrez/eutils/soap/eutils.wsdl:run_eFetch
wsdl:http://eutils.ncbi.nlm.nih.gov/entrez/eutils/soap/eutils.wsdl:run_eSearch
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSPhobius.wsdl:runPhobius
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSPhobius.wsdl:poll
wsdl:http://www.ebi.ac.uk/Tools/webservices/wsdl/WSPhobius.wsdl:checkStatus
soaplab:http://www.ebi.ac.uk/soaplab/services/nucleic_gene_finding.getorf

Table 4: *myExperiment*: LDA with 10 topics, using modules as features.

ing the results. We also considered the *log-likelihood* of the model, a value that describes how well the learned model fits the collection. However, *log-likelihood* did not always correspond to our intuition. Other measures of goodness of fit, e.g., *perplexity*, have been proposed, and are appropriate for generative models like LDA. It would be interesting to develop a principled model selection mechanism that combines measures like *log-likelihood* and *perplexity*, with human judgment.

Category selection and presentation. LDA and frequent itemset mining may both generate more categories than can be presented to the user at any one time. Categories naturally form a partial order, see for example Figure 5 and a discussion at the end of Section 4.1. However, there may still be too many top-level categories to show. This calls for principled ways to select the order in which categories are presented, and to organize categories hierarchically.

Using additional features. In this work we focused on using modules and tags as features. In the future we would like to consider using other features for browsing. For example, *myExperiment.org* workflows often contain titles and text descriptions in addition to tags. Extracting features from text and combining them with the structured features is a promising future direction that we plan to investigate. We would also like to consider how properties of the workflow graphs may be used as features. In this paper we did not attempt to consider pairs of connected modules, or sub-graphs, because that would make the feature space even more sparse. However, we are hopeful that this type of analysis will be possible as larger repositories of scientific workflows become available.

Alternative browsing approaches. Finally, we also plan to consider alternative browsing approaches. Balmin and Curtmola recently proposed to use a construct called the *universal navigational lattice* (UNL) for browsing heterogeneous structured repositories [2]. A UNL encodes all possible ways to group features, and organizes these combinations into a lattice for browsing. We plan to evaluate the appropriateness of this solution in our application domain.

Plans for a quantitative evaluation. In this paper we demonstrated the effectivenss of our methods qualitatively, but did not provide a quantitative analysis. Ultimately, effectiveness of a data exploration solution has to be evaluated

by a user study, which we plan to conduct in the future. A user study may, for example, compare the effectiveness of derived tag-based categories with that of a tag cloud, and measure whether the derived categories make it easier for users to identify workflows of interest. Users may also be asked to make judgments with respect to interestingness, or cohesiveness, of the derived categories. Finally, users may be asked to judge the quality of the hierarchical organization of categories.

Because our proposed solutions aim to ease data exploration, they are based on the assumption that the user may not be fully aware of his information need, and therefore cannot make apriori judgments of result relevance. This is because deciding whether a result is relevant presupposes that the user knows the precise question that he is asking. For this reason quantitative measures such as precision and recall are less appropriate in our setting.

6. RELATED WORK

Our work focuses on data exploration for repositories of scientific workflows. The largest public repository is *myExperiment.org* [19], a collaborative platform for the exchange of scientific workflows and experimental plans. A user may browse *myExperiment.org* by tag (using the tag cloud), or search it with a keyword query. Additionally, workflows may be browsed by author or by type, e.g., *Taverna 1* workflows, *Kepler* workflows, etc. Most recent, most viewed, most downloaded, and most faved workflows are also available for browsing, in sorted order on recency or popularity. The FAQ section of *myExperiment.org* lists "improved search, ranking and faceted browsing" under ongoing work.

While several workflow management systems exist and are gaining popularity, many users still create custom workflows using a variety of programming languages. The Comprehensive R Archive Network (CRAN) [8] supports a repository of over 2300 packages contributed by the users. Users may browse the repository alphabetically by package name, or use CRAN Task Views to browse packages by topic. The Comprehensive Perl Archive Network (CPAN) [7] is an online repository of Perl software and documentation. CPAN implements a search engine for the repository, in which a user specifies a keyword query that is evaluated against one, or all, of provided meta-data fields – modules, distributions, and authors. CPAN also provides a browsable view of the repository by topic.

Recently, several proposals have been made for enabling data exploration in scientific workflow repositories. In [21] the authors propose to use distance-based clustering to organize a collection of *VisTrails* workflows. Unlike in our work, where a workflow may belong to multiple categories defined over a sub-set of features, the approach of [21] assigns each workflow to exactly one cluster, and the clustering is performed in the full feature space.

In addition to browsing, workflow creation may also be viewed as a data exploration task. In this setting, the system may assist the user by suggesting modules to add to a workflow being built, or by providing complete workflows to use as examples. In Scheidegger et al. [22], the authors propose to use provenance metadata collected during the creation of workflows to guide semi-automated workflow development. The user interacts with the system by providing workflow fragments as a query, and the system then retrieves similar workflows and presents them to the user. Here, similarity is computed by considering the history of workflow modifications. Unlike Scheidegger et al. [22], who use *workflow provenance* to suggest similar workflows, Leake and Kendall-Morwick [13] leverage *data provenance* in workflow generation. Their system uses a workflow being constructed as a query, and identifies other workflows that gave rise to similar execution traces. The system then determines which modules used in the execution trace should be suggested as extensions to the query workflow.

Our work is inspired in part by the rich body of literature on faceted search and browsing. Faceted hierarchies are used extensively to organize product catalogs (e.g., `Amazon.com`), scientific articles [11], archaeological finds [18], etc. Faceted search remains an active area of research, and many extensions of the basic model have been proposed, including a faceted query language [17], an extension of the presentation to include statistics other than item counts [3], and dynamic faceted search for information discovery [20, 10]. Dakka and Ipeirotis [9] propose a technique for automatic extraction of facets from text documents that leverages external resources such as WordNet and Wikipedia. Using external information, e.g., hierarchies of modules created by domain experts, is a promising future direction for our work.

We use two technical approaches to discover categories for browsing. The first is frequent itemset mining, specifically, the Apriori algorithm proposed in a seminal paper by Agrawal and Srikant [1]. We gave a detailed description of Apriori in Section 3.1. We also use topic modeling, specifically, Latent Dirichlet Allocation (LDA) proposed by Blei et al. [4]. We describe LDA in detail in Section 3.2.

7. CONCLUSIONS

This paper considered the problem of browsing repositories of scientific workflows. We proposed to use frequent itemset mining and topic modeling, and demonstrated how these techniques may be used in our context. We presented results of an evaluation of appropriateness of these techniques on two real repositories of scientific workflows. Finally, we discussed the conditions under which these data exploration techniques work well, and conditions for which additional techniques may need to be developed, and outlined some directions for future work. We believe that enabling effective data exploration in scientific workflow repositories is essential to the continued adoption of workflows in the scientific community. We also believe that this setting presents the data management community with exciting technical opportunities.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[2] A. Balmin and E. Curtmola. WikiAnalytics: Ad-hoc querying of highly heterogeneous structured data. In *ICDE*, 2010.

[3] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. J. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *WSDM*, 2008.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[5] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *Int. Conf. on Concept. Modeling*, pages 369–384, 2005.

[6] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Managing the evolution of dataflows with vistrails. In *IEEE SciFlow Workshop*, 2006.

[7] The comprehensive perl archive network. `http://www.cpan.org`.

[8] The comprehensive r archive network. `http://cran.r-project.org`.

[9] W. Dakka and P. G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *ICDE*, 2008.

[10] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. M. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM*, 2008.

[11] A. Doms and M. Schroeder. GoPubMed: Exploring PubMed with the GeneOntology. 33, 2005.

[12] KEGG: Kyoto Encyclopedia of Genes and Genomes, 2004. Available at `http://www.genome.ad.jp/kegg/`.

[13] D. B. Leake and J. Kendall-Morwick. Towards case-based support for e-science workflow generation by mining provenance. In *ECCBR*, 2008.

[14] National Center for Biotechnology Information. Online Mendelian Inheritance in Man (OMIM). Available at `www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM`.

[15] National Center for the Multi-Scale Analysis of Genomic and Cellular Networks (MAGNet). geWorkbench. Available at `http://wiki.c2b2.columbia.edu/workbench`.

[16] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. Greenwood, K. Carver, M. G. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(1):3045–3054, 2003.

[17] K. A. Ross and A. Janevski. Querying faceted databases. In *SWDB*, 2004.

[18] K. A. Ross, A. Janevski, and J. Stoyanovich. A faceted query engine applied to archaeology. In *VLDB*, 2005.

[19] D. D. Roure, C. A. Goble, and R. Stevens. The design and realisation of the $\text{my}_{\text{experiment}}$ virtual research environment for social sharing of workflows. *Future Generation Comp. Syst.*, 25(5):561–567, 2009.

[20] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. K. Mohania. Minimum effort driven dynamic faceted search in structured databases. In *CIKM*, 2008.

[21] E. Santos, L. Lins, J. P. Ahrens, J. Freire, and C. T. Silva. A first study on clustering collections of workflow graphs. In *IPAW*, 2008.

[22] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and creating visualizations by analogy. *IEEE Trans. Vis. Comput. Graph.*, 13(6), 2007.

[23] B. Wynar. *Introduction to Cataloging and Classification*. Libraries Unlimited, 8 edition, 1992.