

A First Study on Strategies for Generating Workflow Snippets

Tommy Ellkvist
Linköping University

Lena Strömbäck
Linköping University

Lauro Didier Lins
University of Utah

Juliana Freire
University of Utah

ABSTRACT

Workflows are increasingly being used to specify computational tasks, from simulations and data analysis to the creation of Web mashups. Recently, a number of public workflow repositories have become available, for example, myExperiment for scientific workflows, and Yahoo! Pipes. Workflow collections are also commonplace in many scientific projects. Having such collections opens up new opportunities for knowledge sharing and re-use. But for this to become a reality, mechanisms are needed that help users explore these collections and locate useful workflows. Although there has been work on querying workflows, not much attention has been given to presenting query results. In this paper, we take a first look at the requirements for workflow snippets and study alternative techniques for deriving concise, yet informative snippets.

1. INTRODUCTION

Scientific workflow and workflow-based systems [7, 10, 16, 17, 20] have emerged as an alternative to ad-hoc approaches for documenting computational experiments and design complex processes. They provide a simple programming model whereby a sequence of tasks (or modules) is composed by connecting the outputs of one task to the inputs of another. Workflows can thus be viewed as graphs, where nodes represent modules and edges capture the flow of data between the processes.

Designing workflows is often a time-consuming task and for many applications, workflows become complex and hard to understand. The creation of shared workflow collections has opened the possibility of reusing workflows so that they can be *re-purposed* for new tasks [4, 8, 13, 22]. An interesting example is Yahoo! Pipes [23], where workflows are commonly built by refinement. Having a search engine for these collections is essential to help users locate relevant workflows that can be used as building blocks for new tasks. A search engine must support two important tasks: querying and displaying the results. While there has been work on the former [2, 13–15], the latter has been largely overlooked.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KEYS'09, June 28, 2009, Providence, Rhode Island, USA.
Copyright 2009 ACM 978-1-60558-570-3/09/06 ...\$10.00.

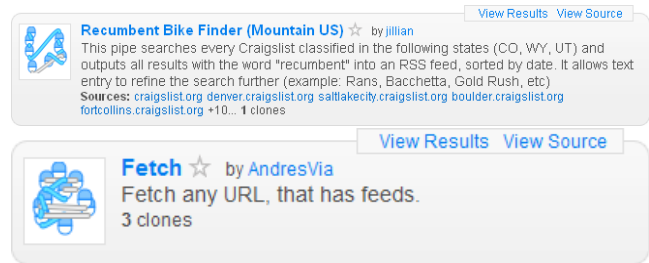


Figure 1: Snippets from Yahoo! Pipes. The quality of snippets depends on the quality of the metadata associated with a workflow. The snippet on the top has an informative description, the one on the bottom does not.

In this paper, we study the problem of displaying results of queries over workflow collections. There are different pieces of information associated with a workflow, including its specification (modules, connections and parameters) and additional metadata, such as a textual description and the workflow creator. To display query results, an important challenge is how to summarize this information. Similar to snippets generated by traditional search engines, workflow snippets can help users more quickly identify *relevant* workflows, without having to inspect their details.

Existing workflow search engines, such as Yahoo! Pipes and myExperiment [9], use metadata associated with workflows to derive snippets (see Figures 1 and 2). As a result, the quality of the snippets is highly-dependent on the quality of the metadata associated with the workflows. For example, while the top snippet in Figure 1 provides a detailed description of the workflow, the one on the bottom does not. Although they also show a thumbnail with a visual representation of the workflows, they do not exploit the actual workflow specification. We posit that by exploiting the workflow specification, we can generate high-quality snippets even when metadata is poor or non-existent.

An important challenge we need to address is how to display *enough* structural information given space constraints. We are faced with conflicting requirements: snippets need to be concise and, at the same time, informative. If snippets are too large (see e.g., Figure 2), only a few will fit on a page, forcing users to browse through multiple pages. If they are too small, users will be forced to examine a potentially large number of workflows. Since workflows can be complex, contain several modules and many more connections, not only it is hard to fit multiple results on a page, but also the complexity of the workflows may confuse the users.

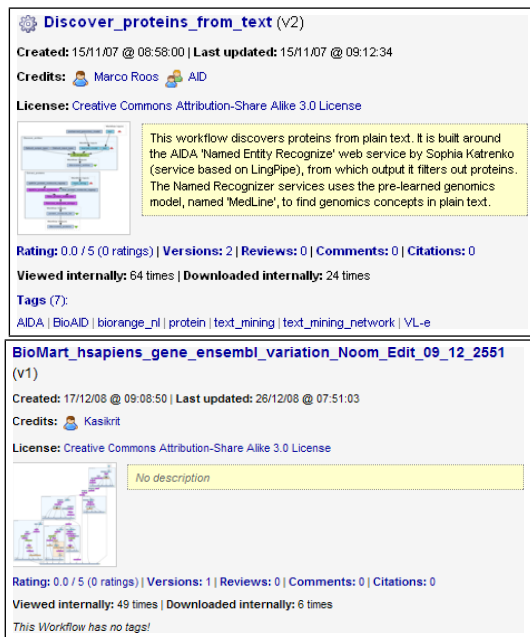


Figure 2: Snippets from myExperiment. While the top snippet contains a detailed description of the workflow and additional metadata, such as popularity and number of downloads, the bottom snippet contains very little information making it hard for one to determine what the workflow actually does.

Contributions. In this paper, we propose and evaluate alternative snippet generation strategies that leverage the information in the workflow graph. Unlike textual documents [18], the structure present in workflows makes the problem of summarization more challenging. Our methods take into account the fact that workflows can be large and dissimilar, and that multiple workflows need to be presented given limited space. Our contributions can be summarized as follows:

- We discuss requirements for generating high-quality workflow snippets;
- We propose alternative techniques for selecting important subsets of the workflow information to be displayed in a snippet; and
- We present a user study where we evaluate the proposed techniques, and compare them against existing workflow snippet generation strategies. Our preliminary results show that quality of the snippets (perceived by the users) improves when structural information is included.

Outline. In Section 2, we define the problem of generating workflow snippets and discuss the requirements for generating high-quality snippets. We describe different strategies for generating and presenting the snippets in Section 3. In Section 4, we evaluate these strategies and discuss our preliminary results. We present related work in Section 5 and conclude in Section 6, where we outline directions for future work.

2. PROBLEM FORMULATION

In what follows, we first provide a formal definition of a workflow and then discuss the requirements for generating workflow snippets.

2.1 Definitions

A *workflow* is a set of partially ordered modules whose inputs include both static parameters and the results of earlier computations. A *parameter* represents a data value. A *module* m is an atomic computation $m : P_I \rightarrow P_O$, that takes as input a set of arguments (input ports P_I) and produces a set of outputs (output ports P_O). The parameters are pre-defined values for ports on a module and can be represented as a tuple $(module_id, port, value)$. In addition *connections* link modules through undefined ports whose values are produced at run time. In a connection $\{(m_i, port_i, m_j, port_j)\}$, the value output on $port_i$ of module m_i is used as input for $port_j$ of module m_j . A set of modules M along with a set of connections C define a partial order relation PO_M on M . This partial order does not contain cycles—the workflow is a directed acyclic graph, or *DAG*—and defines the execution order of modules. Figure 3A shows a connected set of modules. Pre-defined parameters on modules are not shown. Besides the *workflow specification* W , workflows are also associated with a list of annotations A . *Annotations* are triples on the form *subject, key, value* where *subject* is a component of the workflow (e.g., a module or sub-graph), *key* is the type of annotation, and *value* is the actual annotation. Annotations are used to attach metadata to the workflow.

A *workflow collection* C contains a set of workflows. A *search query* consists of a list of query terms Q . Any word w in the workflow specification such that $w \in Q$ is considered a *keyword*.¹ A search query over C returns a *result set* R where $R \subset C$ and $|R| = k$. A *workflow snippet* S provides a summary of a workflow specification. It consists of a *subset of the information in the workflow specification*: $S \subset W \cup A$. **Snippet Generation.** Given a workflow w , we aim to construct a snippet S that is informative and concise. The workflow collection (C), query (Q), and result (R) set are potential sources of information that can be included in the snippet. We introduce a function $FScore: \{C, Q, R, W, S\} \rightarrow c$, which given these information sources, the snippet S and workflow W , outputs a score $c \in \mathbb{R}$, which indicates the usefulness of the snippet. Our goal is to generate a snippet S the highest possible *FScore*.

2.2 Snippet Requirements

Snippets should allow users to browse and determine the relevance of workflows more efficiently, i.e., a user should be able to identify workflows that are relevant to her query by examining the snippets, and should not need to inspect the details of each workflow in the result set.

Huang et al. have studied the problem of generating snippets for XML query results [5]. They outline four criteria to evaluate snippets. Snippets should be: *self-contained*, *representative*, *distinguishable*, and *small*. Below we describe how we map these criteria to the workflow snippets, and in Section 3, we discuss how these criteria are used as the basis to design snippet generation strategies.

Self-contained. A snippet should show the context of keywords in the query. Each module m is associated with a set of keywords $key(m)$ which are extracted from the module name, type, annotations, and parameters. If a query Q matches a module, $key(m) \cap Q \neq \emptyset$, the module m and

¹Although, in this paper we focus on keyword-based queries, our techniques can be extended to support structural queries [13, 14].

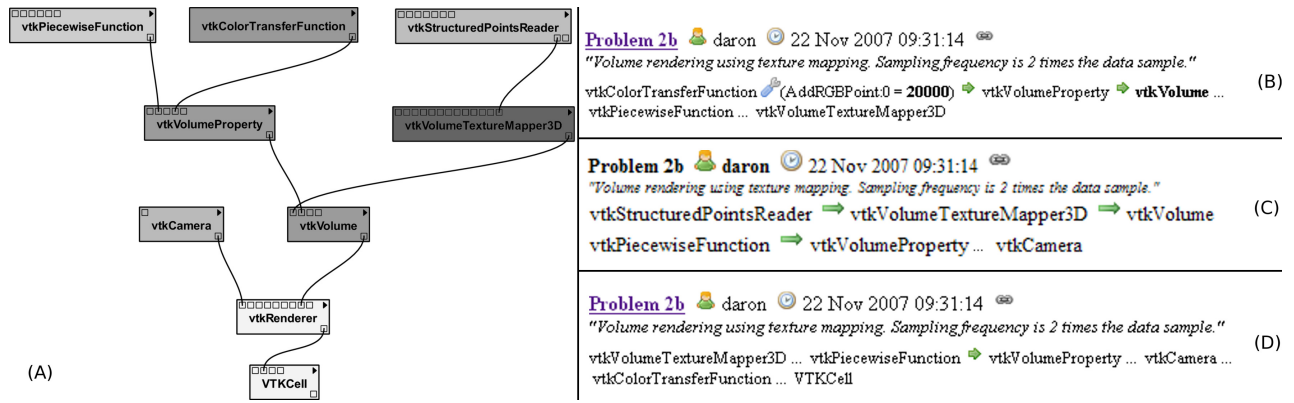


Figure 3: (A) A workflow where darker modules indicate more uncommon modules (IDF-value). (B) A snippet showing the neighborhood matching the query “vtkVolume AND 20000”. The query keywords are highlighted and the keyword neighborhood is displayed as a set of paths. (C) A snippet showing the modules with highest IDF value. (D) A snippet showing representatives for the most important groups.

other modules in its neighborhood should be included in the snippet. Since there can be overlaps of the neighborhoods of modules that match a query, the snippet generation strategy must find the set of modules $M_S \subset C$ that is *most relevant*. **Representative.** A snippet should capture the essence of each workflow—what the workflow actually does. This is analogue to how sentences are selected to represent a document in text snippets. We need to identify the most prominent features in the workflow graph that best represent its semantics, i.e., $M_R \subset M$.

Distinguishable. The difference between any pair of snippets should be visible. To do so, we must find and display the structural differences among the workflows in a result set. In other words, we need to identify the set of most distinguishing modules $M_D \subset M$. This is analogous to clustering text documents and extends the usual representation of snippets to also include information about other workflows in the result set.

Small. Snippets should be compact so that many can fit on a single result page. We do this by fixing the maximum number of modules in each snippet to g .

Given these requirements, a workflow snippet can be constructed as a combination of the relevant modules $S = \{M_S \cup M_R \cup M_D\}$, such that $|S| \leq g$. Below, to understand the implications of applying these different criteria, we examine each individually. We note, however, that in the general case we need a mechanism to weigh them and select the *best* combination of modules.

3. SNIPPET GENERATION

The focus of our work is on generating snippets that exploit the graph structure of workflows. To concisely represent a workflow graph, we propose different summarization strategies that selectively display (or hide) components of workflow graphs. Before presenting different strategies for generating and displaying snippets, we describe two metrics that capture the notion of importance of sub-components of a workflow.

3.1 Structural Importance

In order to select the subsets of a workflow that should be displayed (or hidden) in a snippet, we first need to define a notion of importance. The intuition behind our choice

comes from two empirical observations: modules that are present in a given workflow but that are unusual in the result set (or in the workflow collection) might be of interest to the user—they are a *distinguishing* feature of the workflow. Furthermore, modules that occur together in many workflows represents a semantic entity (a pattern) that can be of interest to the users.

To identify distinguishing modules we apply a measure that is widely used in information retrieval: the Inverse Document Frequency (IDF) [1]. IDF is defined as:

$$idf(m) = \log N/N(m)$$

where $N = |C|$ and $N(m) = |\{W : W \in C \wedge m \in W\}|$. We can then estimate, for each module type, how rare it is in the collection. As an example, Figure 3(A) shows a workflow where darker modules correspond to modules that have a higher IDF value—the modules are more uncommon in the collection.

By measuring module co-occurrence (or semantic proximity), we can find interesting groups of modules. These groups form semantic entities that can be represented in a more compact way, e.g., by collapsing a group into a single module. A similarity measure that fits our need is the Jaccard distance [19]. The semantic distance between module types M_A and M_B can be defined as:

$$dist(M_A, M_B) = 1 - \frac{P(M_A \cap M_B)}{P(M_A) + P(M_B) - P(M_A \cap M_B)}$$

which requires the frequency of M_A in C , the frequency of M_B in C , and the frequency of M_A and M_B occurring together in C . Note that $dist(M_A, M_B) = 0$ when M_A and M_B always occur together, and $dist(M_A, M_B) = 1$ when they never occur together.

3.2 Module Selection Strategies

Below, we explore different strategies for selecting a subset of the modules in the workflow to be included in the snippet. **Query-Neighborhood Strategy.** This strategy identifies important modules M_S in the neighborhood of a set of modules that match the query keywords. We use the following analogy to snippets for text documents: modules represent words, and connected sets of modules correspond of sentences. We first choose all modules M_Q that match the query Q , and traverse their connections to find the neigh-

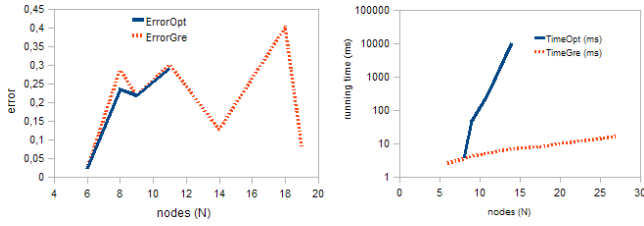


Figure 4: Comparison between optimal and greedy grouping algorithms. Left, difference in total error versus number of nodes. Right, the running time versus number of nodes.

borhood. This can be done through a breadth-first search from the initial set of modules that stops when g modules are found. Since g is small, we select the modules with the highest IDF value first, i.e., the most descriptive modules are selected first. The snippet in Figure 3(B) shows a representation of such a neighborhood split up into paths.

IDF Strategy. The goal of this strategy is to discover a set M_R of modules that are *representative* for a workflow. Using the IDF measure, we can choose the top g modules with the largest IDF value. This strategy is used for the snippet in Figure 3(C). The main advantage of this strategy is that common modules that do not contribute to the understanding of the workflow are selected last. A possible drawback is that the most uncommon modules may occur within the same part of the workflow, which may leave other, potentially important sub-graphs unrepresented. The next strategy addresses this issue.

Grouping Strategy. This strategy attempts to group together modules that co-occur often. This is based on the observation that workflows may contain sub-graphs with specific functionality which can be presented as a single entity, making for a more compact and yet useful representation. We divide the graph into sub-graphs (groups) such that each group M_n is a connected sub-graph of M . We want to find a set of disjoint groups $G = \{M_1 \dots M_g\}$ that covers the whole graph. We use the Jaccard distance to measure the semantic relatedness of specific groups using:

$$MScore(M_n) = \frac{\sum_{m_i, m_j \in M_n} dist(m_i, m_j)}{|M_n|}$$

The score for the workflow is:

$$GScore(G) = \sum_{M_i \in G} MScore(M_i)$$

Testing all possible G reduces to the Exact Cover problem [6], which is NP-complete. We have used a greedy algorithm that, starts with a group for each module, and merges the two adjacent groups that give the best $GScore$ until g groups remains. This approximation reduces the complexity to polynomial time. Figure 4 shows the difference between optimal and greedy versions. The optimal version fails when $|M| > 14$ whereas the greedy scales well. The approximation error of the greedy version is small in most cases, but was not computable for large workflows.

Once we have a set of groups we represent, for each group G , we select the module with the highest IDF in G to be the representative for that group. Another alternative would be to select module with the lowest Jaccard distance, but this leads to the selection of modules that are too common and do not reflect the complexity of the group. The IDF

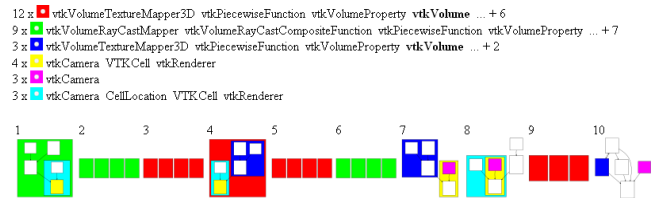


Figure 5: Multiple snippets having common groups. The algorithm identifies shared groups and shows them with colors. A legend showing the meaning of the colors is shown on the top. (Best viewed in color.)

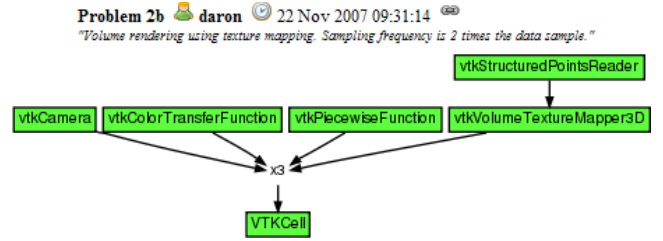


Figure 6: A snippet presented as a partial workflow graph. x_3 indicates that 3 modules have been collapsed.

method, in contrast, often selects uncommon (more specialized) modules and provides more meaningful labels for the groups Figure 3(D) shows an example of a snippet derived by this strategy. Although the groups are not shown explicitly, notice the addition of $VTKCell$ that represents a previously unrepresented part of the workflow.

Difference Highlighting Strategy. This strategy aims to display the differences and similarities among workflows in a result set. Because workflows in a set may share sub-graphs, seeing the differences and similarities may help users obtain a global understanding of the workflows and identify specific features (e.g., modules or groups that occur in many workflows). Since a result set consists of multiple workflows, it is not possible (or even desirable) to show *all* differences, so we focus on identifying the most prominent ones. The strategy works as follows. First, the grouping strategy is used to find common groups. To increase the number of matches, we consider two groups if they contain the same types of modules, disregarding the graph structure. Sub-groups can then be selected that make it possible to match sub-graphs of large workflows with small workflows. This attenuates the effect of oversimplifying large workflows which leads to the hiding of structures that they may have in common with smaller workflows. Based on user feedback (see Section 4), we have designed presentation method for highlighting the group containment, which is illustrated in Figure 5. The snippets have been aligned horizontally to make the differences clearer.

3.3 Snippet Presentation

As Figure 3 (right) illustrates, a textual representation for a snippet can be compact and informative, but structural information is lost since only a few paths of the workflow are shown. As an improvement, we propose to present a dynamic image of the a sub-graph of the workflow that represents the thumbnail, as shown in Figure 6, which uses the same strategy as the snippet in Figure 3C. The idea is to make the graph similar to that seen in a workflow editor,

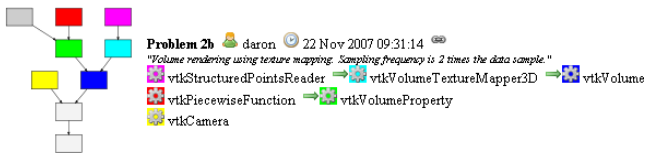


Figure 7: A snippet presented as a graph with a color legend describing the modules. (Best viewed in color.)

thus easier to interpret for experienced users. The disadvantage is that processing time increases—multiple graphs must be dynamically rendered; and additional screen space is needed for each snippet. As a compromise, we designed a snippet where a textual legend is displayed next to the graph (see Figure 7). The graph can then be very small and still contain dynamic information. A problem seen with this approach is that users must keep referring to the legend to understand the graph.

When presenting differences of multiple snippets, user feedback suggests we should not represent each snippet individually, but together, and clearly highlight their differences and similarities. Thus we need a more compact way of presenting multiple snippets. A alternative we have tried is to represent each snippet with a graph consisting of common groups, as shown in Figure 5. Groups that are similar have the same color and differences are represented by white groups. It is then easy to see if workflows are similar in structure. One drawback of this method is that users must refer to the legend to know what each group represents, and groups without a legend are not useful. It is also sometimes not intuitive to spot the differences between workflows; they may have nothing in common; the group order may be rearranged, breaking spatial similarity; and some differences may be hidden, due to lack of space. We intend to explore these issues in future work.

4. PRELIMINARY EVALUATION

In order to evaluate the proposed snippet generation strategies, we performed a user study where we sought both qualitative and quantitative feedback. In our study, we looked at workflows created by students of a visualization course, that make use of the VTK library [21].

Quantitative Feedback. In order to examine the effectiveness of our methods we designed a questionnaire and applied it to six subjects: three were experts and three were knowledgeable users. The questionnaire consisted of three parts. The first part asked users to score different workflow features. The results are presented in Figure 8(a). The workflow *description* was seen as most important component, followed by the *module* types in the workflow. This supports our intuition that what the workflow does is important, whether it is described (in text) or presented by showing workflow specification (the graph). The actual structure, i.e., *connectivity* and *size*, seems less important. Metadata like popularity, *author* of the workflow and creation *time* were seen as somewhat important, but orthogonal to contents.

In the second part users were presented with four workflows which were chosen to reflect differences in size and structure. Each user first chose the six most important modules in each workflow. We then counted how many module types users selected that were common to the modules selected using the IDF and grouping strategies. The resulting scores are shown in Figure 8(b). The mean score for IDF

is 3.4 and the core for grouping is 2.8. The scores are relatively low since not all users selected the same modules. Nonetheless, these scores indicate that IDF is a good measure of importance with the grouping strategy slightly behind. In the second task, users were asked to score four different snippet types: Snippet 1 contained the same information as the Yahoo! Pipes snippet in Figure 1, showing a thumbnail of the workflow; Snippet 2 contained structure presented as text as in Figure 3; Snippet 3 contained thumbnail with legend as in Figure 7; and Snippet 4 contained the workflow sub-graph as in Figure 6. The results are presented in Figure 8(c). Snippet 3 scores best and also contains the most information. Snippet 4 comes second, indicating that it may not be worth to have a bigger snippet at a higher cost. Snippet 2 scores better than snippet 1, indicating that users prefer to have structural information in the snippet.

We note that, overall, the scores seem low. This reflects the scale we used, where 3 = *good* and 5 = *excellent*. But it also suggests that there is room for improvement.

Qualitative Feedback. Preliminary user feedback gave valuable insights. The users found that showing the neighborhood of the query keyword and using IDF to find important modules lead to better snippets. The labeling strategy, on the other hand, was criticized because it splits the structure of the workflow and the contents of the modules. In general, users found it inconvenient to see groupings that lacked labels and whose structure greatly differed from the structure of the original workflow (see Figure 5).

5. RELATED WORK

Current workflow search engines such as Yahoo! Pipes and myExperiment use descriptions and coarse-grained thumbnails in their snippets. Our work extends this to consider fine-grained structural information. The WISE [15] workflow search engine addresses the related problem of showing sub-workflows containing keyword terms. Structural snippets have been explored for XML documents [5]. In [5], Huang et al. propose a set of requirements for snippets that we used as the basis for designing workflow snippet generation strategies. Previous works on snippets for ontologies on the Semantic Web uses semantic proximity of concepts computed with different tools and databases such as Wordnet² to calculate the query neighborhood [11]. They focus on self-similarity and keyword proximity and have a number of important measures that could be useful for workflow snippets and we plan to consider this in future work. Considerable work exist on finding the query neighborhood for text documents [18], our method is analogous but is applied to the workflow graph. Our work was inspired, in part, by approaches to compress network graphs [3]. Although these take into account mainly the topology of the graph, they can be combined with workflow heuristics. Computing the maximum common sub-graph between any two workflow graphs is a related problem [13], but not directly applicable for snippet generation due to its computational complexity. The problem can be simplified by reducing the workflow structure to a set of module labels, or a multiset of labels. This simplification have been shown to yield a good approximation of the workflow structure [12]. We use this idea for finding similar groups. Our grouping approach is related to document snippet clustering [24] but create groups

²<http://wordnet.princeton.edu>

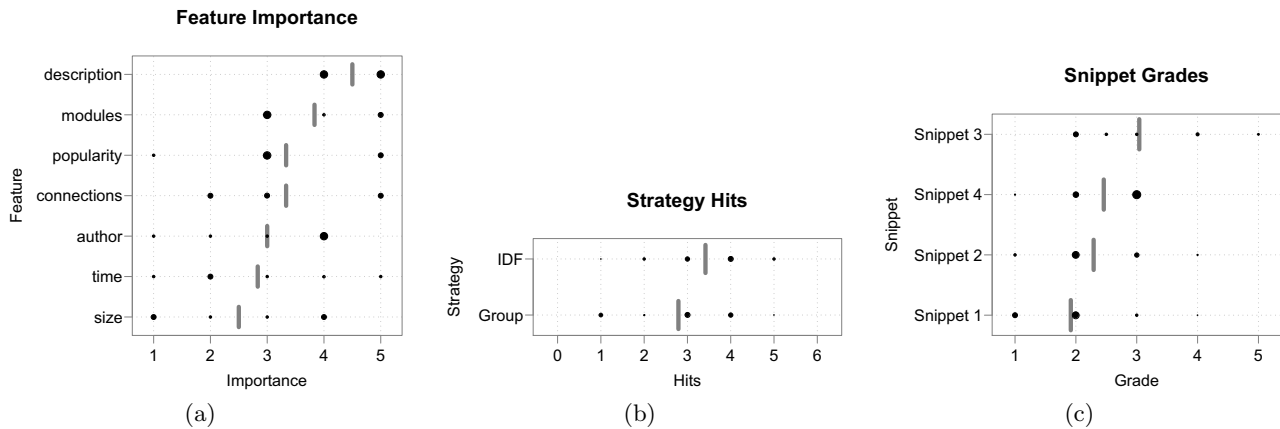


Figure 8: Results of questionnaire. Gray bars represents mean scores. Black circles represent observed scores: larger black circles means more scores on that value. (a) Features that users found important sorted by mean importance. (b) The scores for the IDF and Grouping strategy. (c) User grades for the four snippets sorted by mean value.

as a connected sub-graph. Grouping can be compared with agglomerative clustering, whereas our optimal approach is similar to k-means.

6. CONCLUSIONS

This paper presents a first study on constructing workflow snippets using information from the workflow graph. Our preliminary results show that structural information is useful and conveys important information about the workflow. Our work hints that the simple IDF strategy is preferred to the more involved grouping strategy and that snippets that resemble the actual workflows are preferred by the users. The results also show that there is room for improvement in the quality of workflow snippets, and we plan to investigate additional strategies in future work.

Acknowledgments. This work is partially supported by the NSF, the DOE and an IBM Faculty Award.

7. REFERENCES

- [1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press/Addison-Wesley, 1999.
- [2] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *VLDB*, pages 343–354, 2006.
- [3] A. C. Gilbert and K. Levchenko. Compressing network graphs. In *LinkKDD workshop*, August 2004.
- [4] A. Goderis, U. Sattler, P. Lord, and C. Goble. Seven bottlenecks to workflow reuse and repurposing. *LNCS*, 3729:323, 2005.
- [5] Y. Huang, Z. Liu, and Y. Chen. Query biased snippet generation in XML search. In *SIGMOD*, pages 315–326, 2008.
- [6] R. Karp. Reducibility among combinatorial problems, 85–103. In *Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, NY*, 1972.
- [7] Kepler. <http://kepler-project.org>.
- [8] D. Koop, C. Scheidegger, S. Callahan, J. Freire, and C. Silva. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.
- [9] myExperiment. <http://www.myexperiment.org>.
- [10] The Pegasus Project. <http://pegasus.isi.edu>.
- [11] T. Penin, H. Wang, D. T. Tran, and Y. Yu. Snippet generation for semantic web search engines. In *ASWC*, DEC 2008.
- [12] E. Santos, L. Lins, J. P. Ahrens, J. Freire, and C. Silva. A first study on clustering collections of workflow graphs. In *IPAW*, 2008.
- [13] C. Scheidegger, D. Koop, H. Vo, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007.
- [14] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and re-using workflows with vistrails. In *SIGMOD*, pages 1251–1254, 2008.
- [15] Q. Shao, P. Sun, and Y. Chen. WISE: A Workflow Information Search Engine. Demo Description. In *ICDE*, pages 1491–1494, 2009.
- [16] Swift. <http://www.ci.uchicago.edu/swift>.
- [17] Taverna. <http://taverna.sourceforge.net>.
- [18] A. Turpin, Y. Tsegay, D. Hawking, and H. Williams. Fast generation of result snippets in web search. In *SIGIR*, pages 127–134. ACM New York, NY, USA, 2007.
- [19] C. Van Rijsbergen. *Information retrieval*. Butterworth-Heinemann Newton, MA, USA, 1979.
- [20] VisTrails. <http://www.vistrails.org>.
- [21] VTK. <http://www.vtk.org>.
- [22] X. Xiang and G. Madey. Improving the reuse of scientific workflows and their by-products. In *IEEE ICWS*, pages 792–799, 2007.
- [23] Yahoo! Pipes. <http://pipes.yahoo.com>.
- [24] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*, pages 210–217, 2004.