Proceedings of the Eighth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with SC 2013, Denver, CO, November 17, 2013.

Time-Bound Analytic Tasks on Large Datasets through Dynamic Configuration of Workflows

Yolanda Gil
Varun Ratnakar
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
gil@isi.edu, varunr@isi.edu

Rishi Verma, Andrew Hart,
Paul Ramirez, Chris Mattmann
NASA Jet Propulsion Laboratory
4800 Oak Grove Drive,
Pasadena, CA 91109
{rishi.verma, andrew.f.hart,
paul.m.ramirez,
chris.a.mattmann}@jpl.nasa.gov

Arni Sumarlidason
Samuel L. Park
MDA Information Systems LLC
820 West Diamond Ave., Suite 300
Gaithersburg, MD 20878
arni.sumarlidason@mdaus.com
sam.park@mdaus.com

ABSTRACT

Domain experts are often untrained in big data technologies and this limits their ability to exploit the data they have available. Workflow systems hide the complexities of high-end computing and software engineering by offering pre-packaged analytic steps combined into multi-step methods commonly used by experts. A current limitation of workflow systems is that they do not take into account user deadlines: they run workflows selected by the user, but take their time to do so. This is impractical when large datasets are at stake, since users often prefer to see an answer faster even if it has lower precision or quality. In this paper, we present an extension to workflow systems that enables them to take into account user deadlines by automatically generating alternative workflow candidates and ranking them according to performance estimates. The system makes these estimates based on workflow performance models created from workflow executions, and uses semantic technologies to reason about workflow options. Possible workflow candidates are presented to the user in a compact manner, and are ranked according to their runtime estimates. We have implemented this approach in the WOOT system, which combines and extends capabilities from the WINGS semantic workflow system and the Apache OODT Object Oriented Data Technology and workflow execution system.

Categories and Subject Descriptors

C. Computer systems organization, D.2 Software engineering, D.2.10 Design.

General Terms

Design, Performance, Human Factors.

Keywords

Workflows, semantic workflows, performance, WINGS, OODT.

WORKS'13 November 17, 2013, Denver, CO, USA

http://dx.doi.org/10.1145/2534248.2534257

1. INTRODUCTION

Big data is pushing the boundaries of computing by increasing information available from higher resolution scientific instruments, sensors, business and financial systems, and other data sources. End users with no expertise in big data analytics techniques face many challenges in analyzing their data. Workflow systems make big data analytics more accessible by managing common tasks that big data producers and algorithm developers execute to transform information throughout its lifecycle from data production, to processing/transformation, and ultimately to data distribution [Woollard et al 2008]. Experts can create workflows that represent complex multi-step analytic tasks and share them so that end users can use them with their own data [De Roure et al 2009].

It is possible to design alternative workflows for the same task that have very different performance, particularly for big data. For example, many different workflows can be created using different algorithms to detect popular topics in a large collection of documents (e.g., news articles, tweets, etc). An end user may need a workflow to perform that task within a certain deadline, or that has a desired accuracy. When datasets are very large, and the performance varies widely depending on many metadata characteristics and parameter settings, how can an end user compare and select among possible alternative workflows?

In this paper we describe an approach to enable end users to get workflow solutions that meet their performance requirements, in particular runtime deadlines. We leverage: 1) semantic workflows to automatically generate candidate workflows based on available data analysis algorithms, 2) workflow execution with integrated data and metadata management and provenance recording, and 3) learning predictive performance models from prior workflow executions.

Our approach allows a *workflow designer* (i.e., an expert in big data analytics) to create abstract workflow templates that can be run using different application algorithms, and to provide training data (e.g., sets of inputs and outputs) to be used in a learning phase to create a performance model for each workflow template under different datasets and parameter settings. When a *workflow user* (e.g., an end user with limited or no background on big data analytics) provides a set of performance requirements, the system automatically generates possible candidate specializations of the workflow template and uses the learned performance model to rank those candidate workflows. These ranked candidates are offered to the user, who can choose based on their runtime performance.

We have implemented this approach in WOOT, a system that combines and extends the semantic workflow reasoning capabilities of WINGS [Gil et al 2011a; Gil et al 2011b] and the metadata extraction and provenance tracking capabilities of the Apache OODT Object Oriented Data Technology and workflow execution system [Mattmann et al 2006; Mattmann et al 2009].

The rest of the paper is organized as follows. The next section motivates through examples the needs of end users as they confront the analysis of big data at scale in the face of many alternative algorithms, implementations, and methods. Section 3 surveys related work. Section 4 introduces our approach, defining five key capabilities needed. Section 5 explains the architecture of WOOT that combines and extends WINGS and OODT to achieve those capabilities, walking through examples along the way. Section 6 discusses additional aspects of this problem that would require extensions to our work to date. Finally, we present conclusions and future work.

2. MOTIVATION

Big data requires a range of expertise that very few people have. End users have a deep understanding of their domain and the questions they want answered from the vast amounts of data, but generally do not have the range of skills required to analyze it. Our goal is to empower end users with the ability to analyze their data through the use of workflows.

Consider a social scientist who has access to large amounts of social media data, such as tweets. He is interested in understanding the dynamics of followers based on affinity to topics of tweets. Another example would be a communications student doing a thesis on what groups of teenagers discuss particular topics in social media. Consider also an epidemiologist trying to understand the spread of disease from social media data. Finally, consider a historian who, as suggested by http://programminghistorian.org, has a large set of documents such as multi-year newspaper records[1] or decades worth of daily diaries[2]. She would like to know the topics that were being discussed at any given time and how they changed over the years particularly in response to known historical events.

These are all examples of end users, who may be aware that there are topic modeling techniques developed by natural language experts that they could apply but do not have the necessary skills.

Workflows provide a mechanism to capture state-of-the-art multi-step methods that experts would use for a particular task, and that non-experts can easily reapply. Figure 1 shows an example of a general workflow for topic modeling taken from [Hauder 2011a]. The workflow starts removing stop words (e.g., punctuation) and short words (e.g., "the", "of", "and"), and then converts the data to a format that can be used by a state-of-the-art topic modeling algorithm such as Latent Dirichlet Allocation (LDA) [Blei et al 2003]. This particular workflow was used with little training by high-school students to analyze twitter data [Hauder et al 2011b]. Experts can create workflows and share them with others in repositories [De Roure et al 2009; Ramachandran et al 2009] or as open web objects [Garijo and Gil 2011].

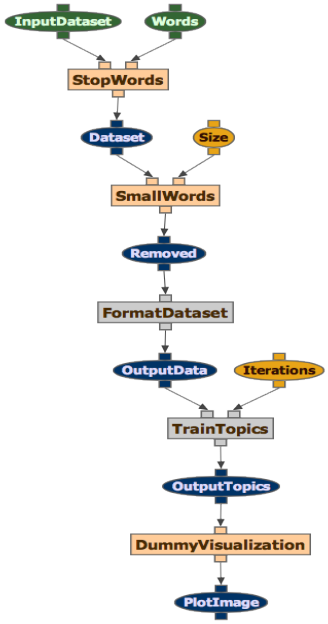

Figure 1: A general workflow for detecting popular topics in a set of documents.

However, many different algorithms and approaches exist for topic modeling step, and many alternative implementations of those algorithms exist, each requiring different parameters and offering efficient performance. Therefore, many different workflows might be possible. Figure 2 shows examples of alternative workflows that instantiate the general workflow in Figure 1, particularly the topic training step. They use different algorithms and implementations for LDA, each one with different parameters and with different performance. An end user would wonder which implementation would give them an answer faster, and what workflows will give the best answer (a "good" quality answer) within the time bounds.

Figure 2(a) shows a workflow (WF-LDA-MALLET) that uses a popular implementation for LDA in the MALLET package [McCallum 2002]. The site indicates:

> *"The MALLET topic model package includes an extremely fast and highly scalable implementation of Gibbs sampling and efficient methods for document-topic hyperparameter optimization."*

An analogous workflow (WF-LDA-TMT) could be built with the TMT software[3], which also implements LDA.

Figure 2(b) shows a workflow (WF-OLDA) that uses online LDA [Hoffman et al 2010], an algorithm for online learning that builds the topic models as it processes documents incrementally:

> *"Online LDA is based on online stochastic optimization with a natural gradient step […]. It can handily analyze massive document collections, including those arriving in a stream."*

There are several implementations of this algorithm leading to different workflows: gensim [Řehůřek 2009] (WF-OLDA-GENSIM) and Vowpal Wabbit [Langford 2011] (WF-OLDA-VW), both in Python.

[1] E.g., http://dsl.richmond.edu/dispatch

[2] E.g., http://historying.org/2010/04/01/topic-modeling-martha-ballards-diary/

[3] http://www-nlp.stanford.edu/software/tmt/tmt-0.4/

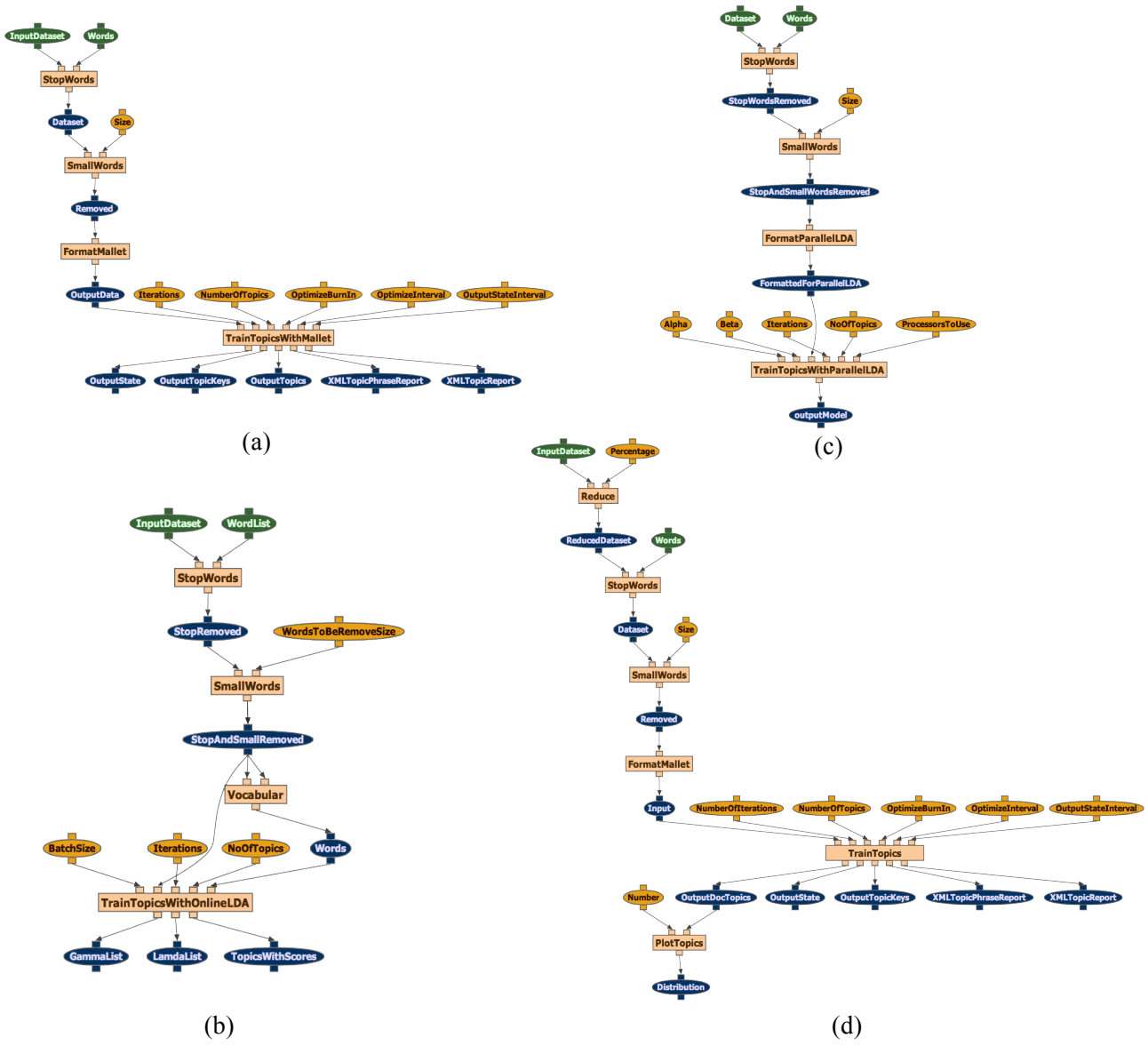

Figure 2: Alternative workflows for topic modeling using LDA: a) Mallet implementation, b) Online LDA, c) ParallelLDA, and d) an LDA implementation that includes a final visualization step and an initial sampling of the input dataset.

Figure 2(c) shows another workflow that uses a parallel version of the LDA algorithm [Wang et al 2009]:

> *"PLDA can be applied to large, real-world applications and achieves good scalability."*

There could be two versions of this workflow: one with an MPI implementation (WF-PLDA-MPI) and a MapReduce implementation (WF-PLDA-MR).

Finally, Figure 2(d) shows a workflow (WF-LDA-VIZ) that first creates a topic model and then generates a visualization of it. In addition, it includes a sampling step at the beginning, which can be used to reduce execution time by reducing the size of the input dataset.

To complicate matters, not only are the above LDA algorithms and implementations very different but they use different input parameters and have a very different performance depending on the value of the parameters. For example, although Mallet LDA and online LDA both have parameters to indicate the number of iterations and desired number of topics, Mallet LDA has another 3 parameters (optimization bounds, optimization interval, and output state interval) that are different from online LDA's other parameter (batch size).

All the algorithms have a parameter to specify the number of iterations of the LDA algorithm. This parameter in particular greatly affects the performance of the algorithm in terms of execution time. Parallel LDA has a parameter to specify the number of processors to use. Below a certain number, performance is likely to suffer from the communication overhead.

In summary, we have described 8 workflow templates for topic detection, all with different algorithms and parameters that affect performance. Suppose a user has a very large dataset, and wants to obtain topics within a certain time bound (e.g., 1 hour). What would be the best workflow for them? This is the problem we aim to address in this paper.

3. RELATED WORK

Retrieving workflows from repositories has been a topic of active research. Some approaches look at matching based on social tags [Goderis 2008], or the shape of the workflow graph [Goderis et al 2008]. In our own work in WINGS, we investigated the retrieval of workflows based on high-level requests, such as finding workflows that generate a desired type of result or process a given type of data [Bergmann and Gil 2012]. Also in our own work on Apache OODT [Mattmann et al 2009], workflows can be retrieved based on a series of dynamic multi-valued metadata that is also used in file cataloging, metadata extraction, and curation and in resource management. This metadata can correspond to workflow status (e.g., FINISHED, EXECUTING; or FAILED); current task wall clock time or workflow wall clock time; workflow id; workflow instance id; and other information. In our own work, and more broadly within the community retrieval of workflows based on performance bounds has not been explored before.

There is a vast literature on performance modeling that is directly relevant to characterizing performance of algorithms (see [Hutter et al 2012] for an overview). Performance modeling relies on collecting performance data for a given algorithm or code with different input datasets and parameter settings, then building a predictive model that can be used to estimate runtime for new datasets. Algorithms are often considered to be "workflow blocks" and their predicted performance based on different input sizes to improve the allocation of resources [Miu and Missier 2012]. In contrast, for this work we want to consider the performance of the entire workflow rather than individual codes. Because our focus is on extending workflow systems to generate and use workflow performance models, our models are quite simple and can be extended in future work to incorporate techniques from this body of work.

Workflow performance is often used to compare across execution infrastructures [Montagnat et al 2010; Vahi et al 2012]. Workflow runtime is one of many metrics that must be taken into account, others can include resource utilization and reliability [Furlani et al 2013; Carrington et al 2005]. Predicting workflow performance is key for resource reservation and resource allocation. All these sophisticated metrics and performance measurements could be used to extend our approach and provide end users with additional tradeoffs.

In prior work, we investigated performance/quality tradeoffs in the context of biomedical image analysis [Kumar et al 2010]. The work focused on the creation of the performance models, which expert high-end computing users would then inspect and based on them decide how to set up parameters for a new dataset. However, there was no interaction with an end user. In addition, the focus was on the selection of parameters, while we focus here on the selection of both algorithms and their implementations in addition to parameters.

4. APPROACH

Our goal is to assist end users with time-bound analytic tasks by recommending workflows that meet their performance requirements in terms of target deadlines. Our approach is to:

1. *Learn workflow performance models*: For every workflow, the system should learn performance models using training datasets of different sizes and characteristics (i.e., metadata) as well as using different parameter settings.
2. *Allow users to specify their requirements and constraints*: Users should be able to specify the desired task and request an answer within a specific time bound.
3. *Automatically generate candidate workflows*: Given the user task and time bound, the system should automatically retrieve all the relevant workflows and instantiate them with possible parameter values.
4. *Rank candidate workflows*: The system should use the workflow performance models to rank the candidate workflows.
5. *Present users with options*: The system should select one or more workflows to run, and present the user with other possible workflows.

Our approach combines capabilities of two complementary workflow systems, WINGS [Gil et al 2011a; Gil et al 2011b] and Apache OODT [Mattmann et al 2006; Mattmann et al 2009], and augments them with new features as follows.

WINGS is a semantic workflow system that facilitates the generation of candidate workflows for user requirements (items 2 and 3 above), since it has a workflow matching engine that can retrieve relevant workflows given a high-level specification of a user's task [Bergmann and Gil 2012] as well as an algorithm to search the space of possible workflow instantiations [Gil et al 2011b]. WINGS supports the specification of abstract workflow templates[4] that include classes of steps (e.g., the workflow in Figure 1 with a topic modeling step) that can be specialized to specific algorithms or implementations (e.g., the workflows in Figure 2, such as 2(c) with topic modeling with Parallel LDA using MPI). WINGS uses workflow reasoning algorithms that take an abstract workflow and automatically generate workflows of executable application codes that can be submitted to a workflow execution engine. For example, WINGS can take the workflow in Figure 1 as input and generate automatically all the candidate workflows shown in Figure 2. While the workflow in Figure 1 is too unspecified to submit to execution, the workflows in Figure 2 are fully specified and can be submitted to a workflow execution engine like OODT.

Apache OODT is a distributed data management and processing framework greatly facilitates learning performance models (item 1 above), since it can extract metadata characteristics upon workflow execution that can be used to do profiling [Mattmann et al 2009], and with distributed execution of workflows with provenance tracking [Mattmann et al 2006]. OODT represents metadata characteristics using OODT's canonical key value pairs that supports a multi-valued metadata representation. OODT also tracks workflow execution and generates provenance records that include execution time for each workflow component and for the overall workflow.

In addition to integrating WINGS with OODT to take advantage of their capabilities, new functionality was needed to learn performance models and rank workflows (item 4 above) and to present the user with options (item 5).

In the next section, we describe our implementation of this approach.

[4] In other research, abstract workflow refers to a workflow with no resources specified to the workflow tasks. Here, abstract workflow refers to workflows with no particular algorithms specified, therefore introducing an additional abstraction layer.

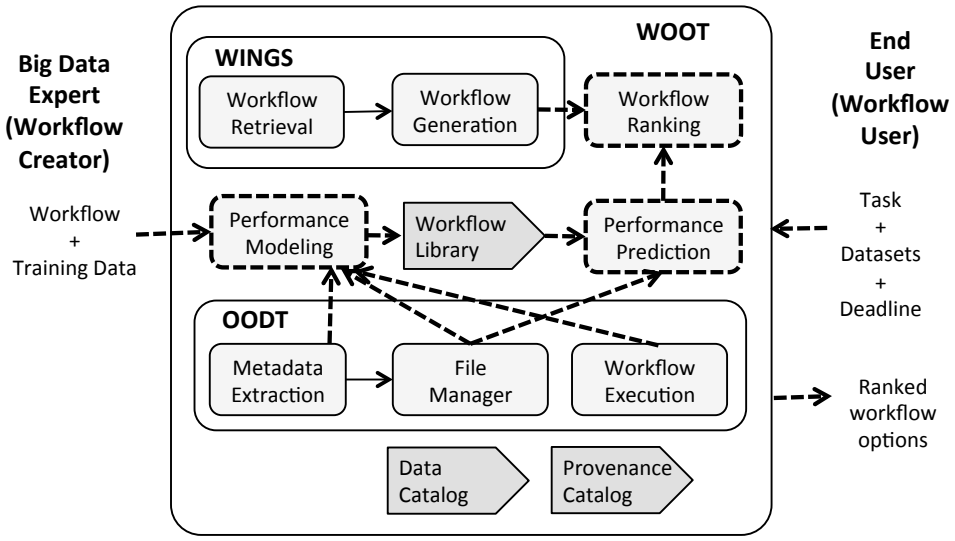

Figure 3: A high-level view of the WOOT system, with the new functionality highlighted with dashed lines. During the learning phase, the system uses training datasets (of different sizes and characteristics) to run workflows in OODT and extract metadata to be used as features to train performance models of the workflows. When the system is in use and the user provides a task and a deadline, WINGS generates relevant workflows (all workflows that include a topic modeling step), ranks them according to their performance, and offers them to the end user as options. The workflows selected are submitted to OODT for execution.

5. WOOT: The WINGS/OODT Workflow Recommender

Figure 3 shows a high-level overview of the architecture of the system, highlighting major WINGS and OODT capabilities used and in dashed lines the new capabilities in WOOT. In the left portion of the diagram, a big data expert provides a set of candidate workflows along with a set of training data and parameters. The system can run the workflows on the training data and record provenance together with metadata and execution times. These form the basis for learning performance models and generating and ranking workflow options allowing selection of an appropriate workflow by the end user as shown on the right of Figure 3.

In the middle of the diagram, the library of candidate workflows is available both to WINGS (at the top of Figure 3) and to OODT (at the bottom of Figure 3). WINGS takes these workflows and the datasets provided by the expert, and generates fully specified workflows that are submitted to OODT for execution. OODT stores provenance metadata about the workflow such as its start/stop wall clock time at a per task level and per workflow level, as well as specific workflow instance metadata (e.g., the input parameters that were provided). This information is stored in the Data Catalog and Provenance Catalog for OODT shown at the bottom of Figure 3.

The Performance Prediction module, shown in the upper right portion of Figure 3, mines the OODT Provenance and Data Catalog to assess workflow performance and ultimately rank the candidate workflows, and to provide this information to the end user in the right of Figure 3.

The next sections describe our implementation of each of the five aspects of our approach listed above.

5.1 Learning Workflow Performance Models

In our approach, a big data expert would create workflows and provide training datasets and parameters that would enable the system to learn workflow performance models.

5.1.1 Creating Workflows

Workflow creators (shown on the left of Figure 3) are big data experts that have experience using the different algorithms available with different datasets and parameters that impact their performance. When creating a new workflow, they are asked to specify some sample datasets of different sizes and characteristics, as well as key parameter values. These datasets will be used to form a comparative model that can be leveraged to predict characteristics of a given workflow, as we describe next.

The semantic features of WINGS workflows are key to our approach. Workflow creators are offered the ability to create abstract workflow templates that include steps that do not refer to specific algorithms or implementations, but instead refer to *classes* of algorithms that carry out a similar task. Figure 4(a) shows an example of such a class hierarchy for topic modeling workflows that an expert would create. The class TrainTopics includes three different algorithms represented as subclasses: Mallet-LDA, Online-LDA, and Parallel-LDA. All three algorithms take an input dataset, a parameter indicating the number of iterations, and output the topic models. These common properties are represented in the class TrainTopics and inherited to each of the three subclasses. Since each algorithm has its own parameters, those are represented in their corresponding subclass as shown in Figure 4(b).

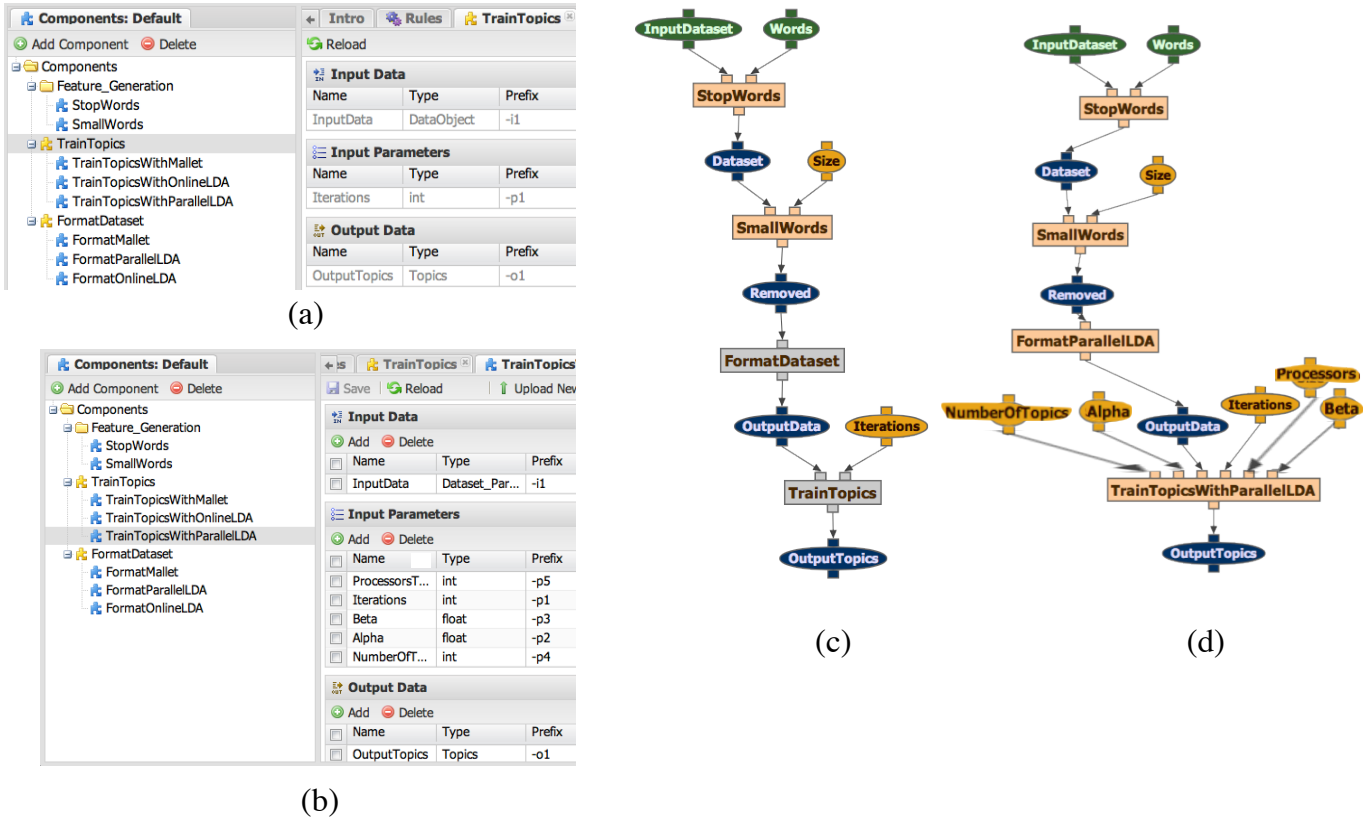

Figure 4: WINGS allows workflow designers to organize workflow components into hierarchies, and use component classes as abstract steps. A component class is shown in (a), where the TrainTopics step represents a class of workflow components that have one input dataset, an Iterations parameter, and an OutputTopics dataset. A workflow component under that class for Parallel LDA is shown in (b), inheriting those three characteristics and having additional parameters such as Processors, Alpha, Beta, and NumberOfTopics. Workflow designers can create abstract workflow templates with the component class TrainTopics as a step, as shown in (c) (same workflow as Figure 1). WINGS automatically generates specializations of that template including the workflow in (d) and other workflows shown in Fig 2(a) and 2(b).

WINGS allows a workflow creator to use component classes as steps in specifying a workflow template. Figure 3(c) shows an example, where one of the steps is TrainTopics, and another step is also an abstract class (FormatDataset). These abstract workflows are presented to the end users, and are automatically specialized with the algorithms available as we describe in Section 5.3 below. An example of a specialized workflow is shown in Figure 4(d) (the same workflow we showed in Figure 2(c)), other workflow candidates include those shown in Figure 2(a)(b).

To train a performance model for the topic modeling workflows discussed here, we used public datasets of document collections containing news items that are widely used in the natural language processing and machine learning communities[5]. The metadata extracted by OODT includes their sizes, which are as follows: R8_train: 3.2MB, R8_test: 1.1MB, R52_train: 4.1MB, R52_test: 1.5MB. We then created additional datasets by selecting random subsets of those documents.

As for parameters, we assume that the experts also provide a selected set of parameter values. For example, in the case of MALLET LDA the parameter that sets the number of iterations is recommended to be set between 1000 to 2000, so the expert can choose a few sample values from that range.

5.1.2 Learning Performance Models

Using the datasets and parameter values provided by the workflow creator, OODT then runs the workflow with all the possible combinations. OODT records all the performance information for each workflow run in a Provenance Catalog. It also extracts and records metadata about the input datasets in a Data Catalog. We created a Performance Modeling module that accesses all this execution information, including metadata of datasets used as well as parameter settings. These form the features (or variables) for learning the performance models. For example:

```
OnlineLDA-Workflow
input1 R8_test   size 1.1MB    numLines 100,000
num-topics 10
num-iterations 1000
optimize-interval 10
optimize-burn-in 20
output-state-interval 0
runtime 160
```

[5] http://csmining.org/index.php/r52-and-r8-of-reuters-21578.html

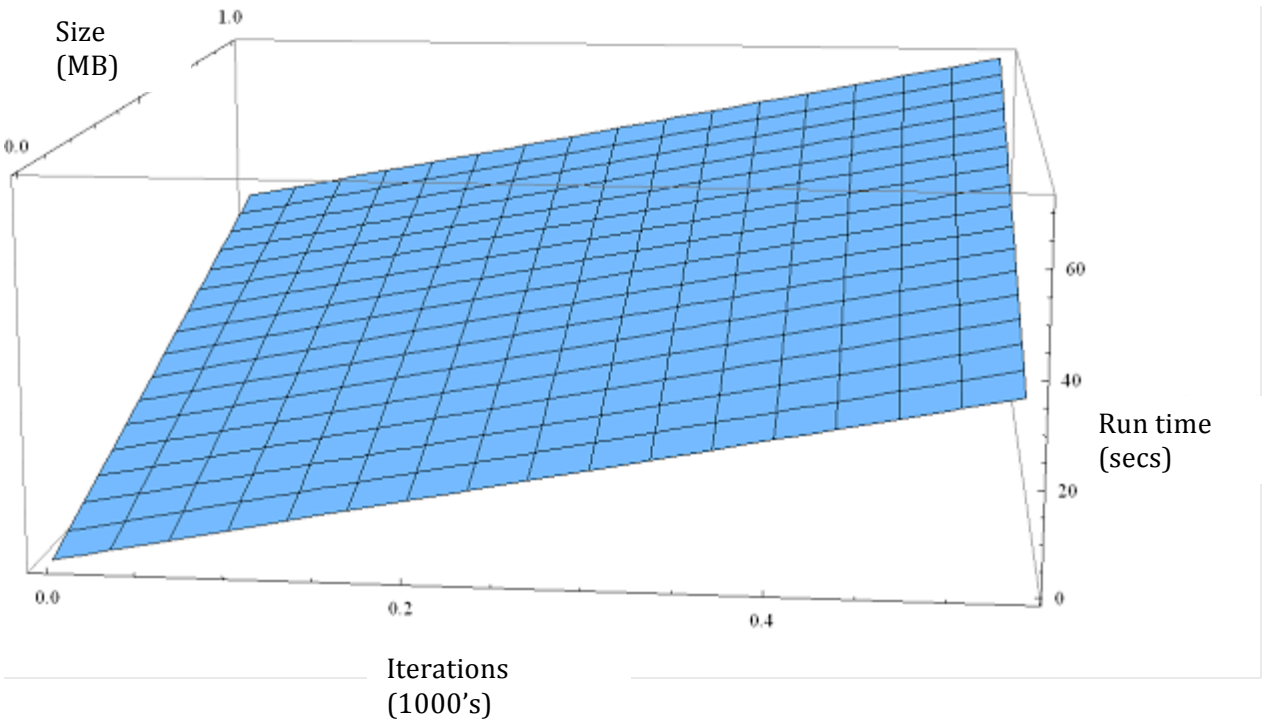

Figure 5: A partial plot of the performance model for the Online LDA workflow shown in Figure 2(b).

Note that the second line has the ID of the input dataset followed by the metadata extracted by OODT represented as key value pairs.

Given this information, the Performance Modeling module learns a performance model by doing a linear regression on the workflow execution data collected from OODT. WINGS will invoke this function when trying to rank candidate workflows for the user, as will be described in Section 5.4.

The Performance Modeling module is very flexible and extensible and operates in the following way: (1) utilizes extension points for incorporating various runtime estimation algorithms, (2) provides run time estimation, among other calculations, via a RESTful web-application tier, and (3) has network connectors to multiple OODT components, such as the File Manager and Workflow Manager.

Figure 5 shows a performance model for the Online LDA workflow. The features that we used included the size of the input file, the number of lines of the input file (this corresponds to the number of documents in these datasets, where one line is used per document), and the values of all the parameters set for the specific run. Shown in the figure are the size and the iterations parameter, the latter affecting runtime more dramatically. Additional metadata properties to train the performance model can be extracted through OODT using Apache Tika [Mattmann and Zitting 2011], such as the number of distinct words, the language of the file, the file format (html, plain text, etc).

5.2 User Request

Workflow users (shown on the right of Figure 3) are end users who have a particular data analytic task at hand, but do not have the analytic expertise of a workflow creator as discussed in the previous section. They would be presented with a collection of abstract workflow templates for different tasks. For example, for the task of topic detection they could be shown the workflow in Figure 1 as a starting point,. Other tasks could include document clustering and document classification; we discuss such collections of text analytics tasks in [Hauder et al 2011a]. It is easy for end users to select workflows based on the tasks that they performed, as shown in [Hauder et al 2011b].

The end user specifies the dataset that they want to analyze, and a time bound. The user can leave the parameters unspecified. The parameters and algorithms that are possible are automatically selected by the system, as we describe next.

5.3 Automatic Generation of Candidate Workflows

Given an abstract workflow template and input datasets provided by the end user, WINGS can automatically generate candidate workflows by performing a search through the space of possible workflows whose steps are specific algorithms that are consistent with that abstract workflow. An overview of the algorithm is given in [Gil et al 2011a], and a detailed description can be found in [Gil et al 2011b]. We briefly summarize it here. First, WINGS specializes the workflow steps by replacing component classes with the possible subclasses, each generating a branch in the search for candidate workflows. Then, WINGS assigns values to all unspecified parameters. Any workflow that is fully elaborated through this search can be submitted to OODT for execution. But first, all these candidates are presented to the end user as we discuss next.

5.4 Ranking Candidate Workflows

For each candidate workflow, the Workflow Ranking module requests an estimate of the workflow runtime from the WOOT

Performance Modeling components described in Section 5.2. The request would be given as follows:

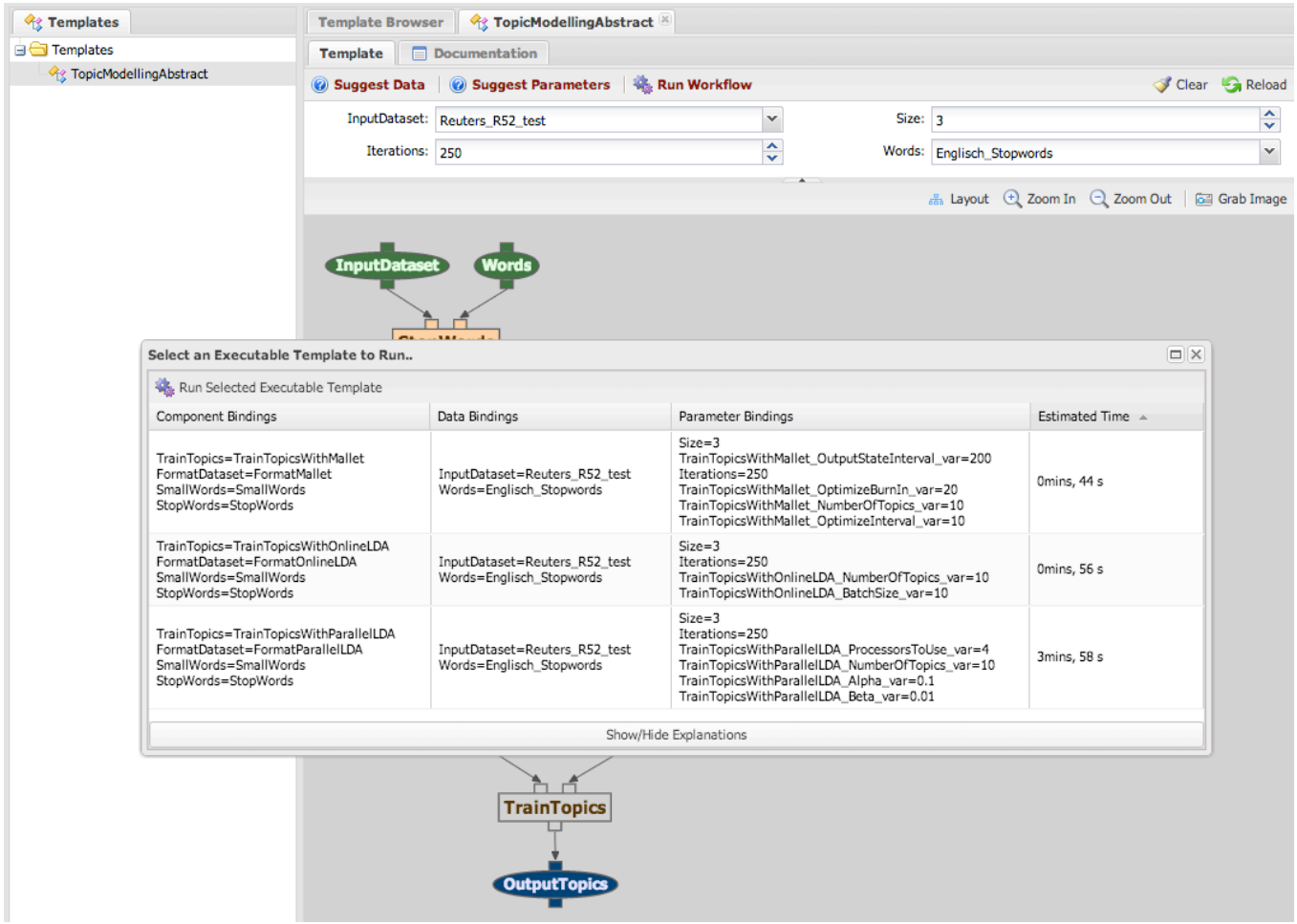

Figure 6: Once the end user selects the abstract workflow template shown in Figure 3(c), the system presents different alternative workflows that use different algorithms and have different runtimes. The user can select the faster one, or select an algorithm that they recognize and prefer. The system could show other estimates in addition to runtime, such as expected accuracy of the answer, general reliability, and other criteria relevant to an end user.

```
OnlineLDA-Workflow
input1 R52_test
num-topics 10
num-iterations 1000
optimize-interval 10
optimize-burn-in 20
output-state-interval 0
```

The Performance Modeling component would return:

```
OnlineLDA-Workflow
input1 R52_test    size 1.5MB    numLines 135,000
num-topics 10
num-iterations 1000
optimize-interval 10
optimize-burn-in 20
output-state-interval 0
runtime-estimate 185
```

Note that the system requested from OODT the metadata needed, including the size and number of lines in the input file. Those metadata properties and values, together with the parameter values, are used to generate the runtime estimate returned in the last line.

5.5 Presenting Users with Workflow Options

Figure 6 illustrates how the system presents workflow options to the end user. Each line represents a possible workflow candidate, and can be selected for execution. The workflow options are not shown as a dataflow graph as workflows are typically shown, since they all share the same dataflow graph represented by the abstract workflow template. To highlight the differences between the alternative workflows, the system shows for each workflow candidate a comparative view in four columns: 1) the particular algorithms that would be used for each step, 2) the input data, 3) the parameter values, and 4) the estimated runtime. The workflow candidates can be sorted according to runtime. The user then selects one or more workflows for execution, which are then submitted to OODT.

The workflow candidates shown in Figure 6 correspond to the 3 LDA algorithms discussed in Section 2, and for each of them the system is suggesting the best parameter values available. Note that our datasets used here are relatively small, but if they were larger they would have dramatically different runtime estimates. This is also the reason why the parallel LDA algorithm has the

worst performance, since the parallelization creates overhead and typically is not an efficient way to process a small dataset.

6. CONCLUSIONS AND FUTURE WORK

In order to enable end users of big data systems to find solutions that suit their needs given a task and a deadline, we presented an approach and implemented system that automatically generates alternative workflow candidates and presents them to the user in a rank order according to performance estimates. The system generates these estimates based on workflow performance models, and uses semantic technologies to generate workflow options. We implemented this approach in the WOOT system, which combines and extends capabilities from the WINGS semantic workflow system and the Apache OODT Object Oriented Data Technology and workflow execution system.

Current limitations of our approach are topics for future work. We describe here three areas of future work: 1) incorporating solution quality estimates, 2) a more sophisticated interaction with the end user, and 3) improving performance models.

An important capability that we could provide is to compare workflows along dimensions other than runtime. These could include domain-specific comparative assessments across workflow options. For example, an extension that would be easy to do in our framework is to allow the Workflow Ranking module to estimate the quality of the solutions. The creator of each workflow would be asked to provide an assessment of the quality of the workflow output as a function of the parameters of the workflow and other metadata. This could be written as a set of rules for the workflow, and they have to be designed in such a way that each set of parameter values leads to a single quality assessment.

For example, for the workflow WF-LDA-MALLET in Figure 1(a) the following rules could be provided:

```
If the number of iterations is less than 1000,
    then the result quality is LOW.

If the number of iterations is more than 1500,
    then the result quality is HIGH.

If the number of iterations is more than 1000
    and less than 1500, then the result
    quality is MEDIUM.
```

Therefore, each workflow instance would have an associated quality estimate and an associated runtime estimate, giving users the ability to explore performance/quality tradeoffs. Such rules for the alternative algorithms are highly domain specific, but they represent knowledge that is very familiar to big data experts who have run the algorithms many times themselves and assessed the relative quality of the results.

We are also exploring alternative quality estimate algorithms based on Bayesian statistics [Winkler 2003]. In this extension, a workflow designer would also provide an assessment of workflow quality (HIGH, MEDIUM, LOW). The system could use the metadata of the input datasets (e.g., size) as features and create a conditional probability distribution over the space of features, such as:

```
P(HIGH|data size<1Gb) = 0.75
P(MEDIUM|data size<1Gb) = 0.2
P(LOW|data size<1Gb) = .05
..
```

The system would take in the above conditional probability distribution as workflow designer input, then execute a Bayesian inference/selection algorithm to combine the conditional probably information into an overall probability for a workflow's quality, given its associated metadata, input, and then this would yield a ranking for the set of candidate workflows.

Another dimension of improvement for our system is the interaction with the user, which could be made more sophisticated. For example, if the user gives a time bound that is not possible with their data and parameters selected, then the system could explain how far the time bound is from what the user needs and suggest choosing a very different abstract workflow for a similar task but perhaps not as thorough. Another possibility would be to show the end user N qualitatively different workflows that used very different algorithms and had very different performance estimates. This would give the user a broader range of options to choose from.

Users could also choose more than just one workflow to submit for execution. Several could be submitted, and the user could look up results as they come in. This would give the user more flexibility and range in the solutions obtained and in the time to solution.

Another extension would be to improve the system performance and confidence as more workflows are run. Once a workflow has been executed, the system would have the predicted runtime and the actual runtime. Clearly it could use the actual runtime to improve its performance model for that workflow. In addition, it could use both values to create a measure of its confidence on the runtime estimates for the workflow and present those to the user as an indication of uncertainty.

We could also create more accurate and refined performance models building on prior work [Miu and Missier 2012; Montagnat et al 2010; Vahi et al 2012; Furlani et al 2013; Carrington et al 2005; Hutter et al 2012]. More refined performance models could take into account dynamic factors such as network latency, queue wait time, and availability of resources required to run a given workflow. Conversely, those frameworks could use our approach to include additional features for the performance estimates, based on the metadata properties that we use.

7. ACKNOWLEDGMENTS

We thank the WINGS and Apache OODT teams for their support of this work. We gratefully acknowledge support from the US Defense Advanced Research Projects Agency (DARPA) with award FA8750-13-C-0016, and from the US Air Force Office of Scientific Research (AFOSR) with award FA9550-11-1-0104.

8. REFERENCES

[1] Bergmann, R.; and Gil, Y. “Similarity Assessment and Efficient Retrieval of Semantic Workflows.” Information Systems Journal, . 2012.

[2] Blei, D., Ng, A., and M. Jordan. “Latent Dirichlet Allocation.” Journal of Machine Learning Research, 3, pp 993–1022, January 2003.

[3] Carrington, L. C. et al. “How Well Can Simple Metrics Represent the Performance of HPC Applications?”, Proceedings of the 2005 ACM/IEEE conference on Supercomputing, 2005.

[4] De Roure, D; Goble, C.; Stevens, R. “The design and realizations of the myExperiment Virtual Research Environment for social sharing of workflows”. Future Generation Computer Systems, 25 (561-567), 2009.

[5] Furlani, T. R., Jones, M. D., Gallo, S. M., Bruno, A. E., Lu, C., Ghadersohi, A., Gentner, R. J., Patra, A., DeLeon, R. L.,

von Laszewski, G., Wang, F., and A. Zimmerman. Performance metrics and auditing framework using application kernels for high-performance computer systems. Concurrency and Computation: Practice and Experience, 25(7), 2013.

[6] Garijo, D.; and Gil, Y. "A New Approach for Publishing Workflows: Abstractions, Standards, and Linked Data." Proceedings of the Sixth Workshop on Workflows in Support of Large-Scale Science (WORKS'11), held in conjunction with SC 2011, Seattle, WA, 2011.

[7] Gil, Y., Groth, P., Ratnakar, V., and C. Fritz. "Expressive Reusable Workflow Templates." Proceedings of the IEEE e-Science Conference, Oxford, UK, pages 244–351. 2009.

[8] Gil, Y.; Deelman, E.; Ellisman, M. H.; Fahringer, T.; Fox, G.; Gannon, D.; Goble, C. A.; Livny, M.; Moreau, L.; and Myers, J. "Examining the Challenges of Scientific Workflows." IEEE Computer, 40(12), 2007.

[9] Gil, Y.; Ratnakar, V.; Kim, J.; Gonzalez-Calero, P. A.; Groth, P.; Moody, J.; and Deelman, E. "WINGS: Intelligent Workflow-Based Design of Computational Experiments." IEEE Intelligent Systems, 26(1). 2011.

[10] Gil, Y.; Gonzalez-Calero, P. A.; Kim, J.; Moody, J.; and Ratnakar, V. "A Semantic Framework for Automatic Generation of Computational Workflows Using Distributed Data and Component Catalogs." Journal of Experimental and Theoretical Artificial Intelligence, 23(4), 2011.

[11] Goderis, A. "Workflow Re-use and Discovery in Bioinformatics." Ph.D. thesis. University of Manchester, 2008.

[12] Goderis, A., Li, P., Goble, C. "Workflow discovery: the problem, a case study from e-science and a graph-based solution." International Journal of Web Services Research 5, 2008.

[13] Hauder, M., Gil, Y. and Liu, Y. "A Framework for Efficient Text Analytics through Automatic Configuration and Customization of Scientific Workflows". Proceedings of the Seventh IEEE International Conference on e-Science, Stockholm, Sweden, December 5-8, 2011.

[14] Hauder, M.; Gil, Y.; Sethi, R.; Liu, Y.; and Jo, H. "Making Data Analysis Expertise Broadly Accessible through Workflows." Proceedings of the Sixth Workshop on Workflows in Support of Large-Scale Science (WORKS'11), held in conjunction with SC 2011, Seattle, WA, 2011.

[15] Hoffman, M., Blei, D., and F. Bach. "Online Learning for Latent Dirichlet Allocation." NIPS, 2010.

[16] Hutter, F., Xu, L., Hoos, H. H., and K. Leyton-Brown. "Algorithm Runtime Prediction: The State of the Art". Available from arXiv:1211.0906.

[17] Kumar, V.; Kurc, T.; Ratnakar, V.; Kim, J.; Mehta, G.; Vahi, K.; Nelson, Y. L.; Sadayappan, P.; Deelman, E.; Gil, Y.; Hall, M.; and Saltz, J. "Parameterized Specification, Configuration, and Execution of Data-Intensive Scientific Workflows." Cluster Computing Journal, 13(3), 2010.

[18] Langford, J. Vowpal Wabbit. https://github.com/JohnLangford/vowpal_wabbit/ 2011.

[19] Mattmann, C., Crichton, D., Medvidovic, N., and Hughes, S. "A Software Architecture-Based Framework for Highly Distributed and Data Intensive Scientific Applications." Proceedings of the 28th International Conference on Software Engineering (ICSE06), pp. 721-730, Shanghai, China, May 20th-28th, 2006.

[20] Mattmann, C. A., et al. "A reusable process control system framework for the orbiting carbon observatory and NPP Sounder PEATE missions." Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), 2009.

[21] Mattmann, C. A., and J. Zitting. "Tika in Action." Manning Publications, 2011.

[22] McCallum, A. K. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[23] Miu, T. and P. Missier. Predicting the Execution Time of Workflow Activities Based on Their Input Features, Proceedings of the Seventh Workshop on Workflows in Support of Large-Scale Science (WORKS'12), held in conjunction with SC 2012.

[24] Montagnat, J., Glatard, T., Reimert, D., Maheshwari, K., Caron, E., and F. Desprez. "Workflow-based comparison of two Distributed Computing Infrastructures." Proceedings of the Fifth Workshop on Workflows in Support of Large-Scale Science (WORKS'10), New Orleans, LA, 2010.

[25] Ramachandran R, Movva S, Conover H, Lynnes C. Talkoot Software Appliance for Collaborative Science. IEEE International Geoscience & Remote Sensing Symposium, 2009.

[26] Řehůřek, R. gensim. http://radimrehurek.com/gensim/. 2009.

[27] Vahi, K., Harvey, I., Samak, T., Gunter, D. K., Evans, K., Rogers, D., Taylor, I., Goode, M., Silva, F., Al-Shakarchi, E., Mehta, G., Jones, A. and E. Deelman. "A General Approach to Real-Time Workflow Monitoring." Proceedings of the Seventh Workshop on Workflows in Support of Large-Scale Science (WORKS'12), 2012.

[28] Wang, Y., Bai, H., Stanton, M., Chen, W., and E. Y. Chang. "PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications." AAIM, 2009.

[29] Winkler, R. L. "An Introduction to Bayesian Inference and Decision. Probabilistic Press, 2003.

[30] Woollard, D.; Medvidovic, N.; Gil, Y.; and Mattmann, C. "Scientific Software as Workflows: From Discovery to Distribution." IEEE Software, 25(4):37-43. 2008.