A Framework for Efficient Data Analytics through Automatic Configuration and Customization of Scientific Workflows

Matheus Hauder
Institute for Software and Systems Engineering
University of Augsburg
Universitätsstraße 6a
Augsburg, D 86135
Email: hauder@student.uni-augsburg.de

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

Yan Liu
Computer Science Department
University of Southern California
941 Bloom Walk
Los Angeles, CA 90089-0781
Email: yanliu.cs@usc.edu

***Abstract*—Data analytics involves choosing between many different algorithms and experimenting with possible combinations of those algorithms. Existing approaches however do not support scientists with the laborious tasks of exploring the design space of computational experiments. We have developed a framework to assist scientists with data analysis tasks in particular machine learning and data mining. It takes advantage of the unique capabilities of the Wings workflow system to reason about semantic constraints. We show how the framework can rule out invalid workflows and help scientists to explore the design space. We demonstrate our system in the domain of text analytics, and outline the benefits of our approach.**

I. Introduction

Advanced data analytics skills require that scientists have knowledge beyond understanding different algorithms and statistical techniques. State-of-the-art data analytics often involves multi-step procedures with sophisticated techniques such as effective feature selection methods, algorithm portfolios and ensembles, and cross-validation. For example, a basic protein secondary structure prediction framework includes sequence analysis, feature extraction, and classification as well as post-processing steps. Workflows are an ideal paradigm for capturing and sharing such complex analysis methods [1], [2] since they can be assembled from individual software tools, then shared and reused as end-to-end methods [3].

A scientist working on the analysis of a dataset must explore many possible methods or combinations of algorithms until an appropriate one is found. A workflow framework facilitates this process by supporting *workflow modifications*, that is, taking an existing workflow and adding/removing steps to create a new workflow [4], [5]. For example, a workflow with a step implemented as a hierarchical clustering algorithm could be easily modified by replacing that step by a k-means clustering algorithm. Currently, these modifications have to be designed and carried out manually by the scientists, with little assistance from the system.

This exploration of the design space of workflows remains a very laborious process. First, many combinations of algorithms may not be compatible with one another. Scientists have to take into account myriad of constraints in order to track which algorithm combinations are valid. Second, some algorithms may not be appropriate for the dataset at hand or require specific data pre-processing steps. Moreover, the best algorithms for a dataset may not be the best for another dataset. For example, a feature selection strategy may not work for a similar dataset with different statistical characteristics. Third, focusing on alternative algorithms for a step may not be worthwhile because other steps are more influential. For example, a good approach to the selection of features to train a classifier is often more important than the type of classifier being used.

The goal of our work is to enable scientists to explore the design space of complex analytic methods in a more systematic and efficient manner. We use workflows as a paradigm to capture complex end-to-end analysis methods. We build on the Wings workflow system [6], which uses semantic workflow representations that capture the requirements and semantic constraints of individual steps and datasets explicitly [7], as well as workflow reasoning algorithms to generate possible combinations of workflow components systematically and to validate them [8]. Our approach is novel because it supports:

- **Easily configurable abstract workflows:** Our semantic workflows can represent abstract steps that can be specialized by many software components. Those components may reflect different algorithms as well as alternative implementations with varying performance and run-time requirements. Each algorithm implementation may include constraints regarding the type and characteristics of data that they are appropriate for. Workflows represent general methods, and are easily configured as executable workflows by specializing any abstract components.
- **Efficient experimentation with workflow specializations:** Our workflow reasoning algorithms can automatically generate possible specializations of components, and reject any specializations that violate constraints of

individual steps. The system is given a workflow with abstract steps and a dataset, and can generate all valid specializations that can be submitted for execution.

The main contribution of this paper is the demonstration of this approach with an implemented framework for text analytic tasks that captures state-of-the-art methods as workflows and enables efficient experimentation with different algorithms and datasets. This framework includes: 1) abstract workflows for both supervised classification and unsupervised clustering, 2) more than 50 components for text analytics, data mining, and machine learning, and 3) several commonly used datasets in the document classification research community. The framework makes it very efficient for a user to learn a new method as a workflow, to experiment with different algorithms and to quickly understand the tradeoffs of choosing different algorithms to analyze a new dataset. Our framework can easily be applied to other scientific data analysis tasks. The Wings workflow system and the text analytics framework are available for download under an open source license from http://wings.isi.edu.

The paper begins with an overview of our motivation and our approach, illustrating the main concepts with examples from the domain of text analytics. Next, an implementation of this approach is described using the Wings workflow system, including the workflows, the software components, and the semantic reasoning used in the workflow system. Finally, we illustrate the advantages of our framework and show experimental results of applying the framework for text analytics tasks in real world datasets.

II. Motivation

Our main motivation is to help scientists to carry out experiments more efficiently, addressing issues that have broader implications in education of novice researchers and students, and in fostering cross-disciplinary work.

Scientists working on data mining and machine learning tasks should be able to design workflows which are appropriate for their own datasets in an efficient manner. Scientists often speak about the data deluge, but not so much about the software deluge. New algorithms and software packages are published constantly, which is difficult for scientists to keep up with the state of the art. Experts can read and be aware of the latest algorithms, but currently do not have a practical means to obtain hands-on experience with them because they require a large investment of effort. We envision the workflow framework of text analytics serves as a aid for scientists to sustain learning as a long-term activity throughout a professional career, enabling experts to keep up with research innovations in an easier, time-efficient, and hands-on manner.

Novice researchers working on data mining for scientific discovery need to be extremely careful to follow correct procedures rigorously. For example, bioinformatics is an actively pursued area in which researchers utilize machine learning and data mining algorithms to reveal significant insights from biology data and to make predictions on structures or functions of unknown proteins/genes. Many classes on bioinformatics teach student basic analysis algorithms, such as sequence alignment, protein structure prediction algorithms, or graph-structure learning algorithms. However, the tradeoffs between different algorithms and their impact on the overall accuracy of the system depend on the application and the dataset. For example, in protein structure prediction, one key step is to find sequences similar to the target protein with a specific similarity cutoff. The subsequent prediction performance is not only determined by the classifiers applied, but also significantly influenced by (more significantly) the similarity cutoff, which is difficult to decide for novice researchers. Each data analytic task in bioinformatics can be associated with an appropriate workflow design process, and facilitating this try and test process is extremely important in order to achieve better prediction performance and even more important to gain correct scientific results.

Our system can also help students to acquire practical skills. Students may take courses about statistical techniques and machine learning algorithms but rarely have the resources to learn in practical settings. For example, in text analytics the prediction accuracy of text classifiers on one dataset can differ 1-10 % depending on how the unstructured texts are converted to feature vectors of words and how many features are selected. In contrast, once the data is preprocessed, the difference between which classifier (whether a support vector machines or Naive Bayes) is applied on the same feature vector is only 0.5-5 %.

Researchers in other disciplines can use our framework to acquire expertise in a new area. For example, social science researchers have access to vast amounts of text data on the web (blogs, microblogs, mailing lists, etc.) that can be mined, but often lack text analytic skills. Similarly, there are reams of text data available in mobile devices and other human-computer interfaces [9]. Developing an appropriate setup for any real problem (e.g., email prioritization) requires good understanding of the state-of-the art methods in that domain (e.g., text analytics), placing a barrier for students or researchers who do not have easy access to that expertise. Moreover, significant software infrastructure needs to be deployed in order to apply and observe different techniques in a real problem domain, requiring substantial investment often in the order of months or years which deters students and researchers in other areas from attempting to do so. Setting up this infrastructure requires programming skills, making it infeasible for students and researchers without significant computer science background.

Finally, our framework can also target researchers that have developed initial data mining applications and are seeking to improve the performance of their application. A good example from our prior work is compiler optimization, where the use of machine learning techniques is being investigated in order to rapidly customize optimizations to new computer architectures that come out every few months. In a recent survey of this research area, we found that most of the work focuses on decade old techniques [10]. Lowering the cost of learning text analytics skills by easily experimenting with different machine learning algorithms would enable new levels of performance

in many application areas.

III. Approach

We have developed a framework for capturing complex text analytic processes and for efficient exploration of different algorithms that can be used as steps in those processes. We achieved this by applying scientific workflows [1] and used a workflow management system. This section describes our framework at a high level, next section describes our implementation of this framework in detail.

A. Flexibility of workflow representations

From a high level perspective document classification consists of a number of computation steps. These steps usually include document preprocessing, feature selection, training and classification. A common set of steps is illustrated in the workflow in Figure 1. In general, a correlation score is computed for each word, followed by a second step that selects the words with the best score as features for the machine learning algorithm. This is crucial to improve the classification results and to minimize the size of the training set.

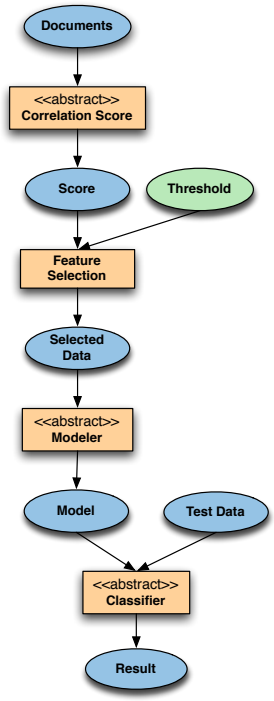


Fig. 1: Exemplary workflow with abstract computations

The algorithms for each step of this workflow are complex, and are often already implemented and available as open source software. Therefore it is important to support the following capabilities:

- **Abstract steps:** Steps in the workflow that can be replaced by other algorithms are represented as abstract steps or components. The correlation score, modeler and classifier in Figure 1 are examples of abstract components. Every abstract step can be specialized by one of several concrete executable steps. These concrete components have exactly the same interfaces and do not overlap with those from other abstract components.
- **Reusing existing software:** A component-based approach [11] enables the reuse of software in a workflow management system. This is useful because there are many existing implementations of algorithms in different languages.
- **Metadata information:** The information about the type of data necessary for a computational step and the requirements it has to fulfill is represented as the metadata information. This information is used to validate the workflow and create provenance information for the executed experiments.

Note that other work defined abstract workflows as those that do not contain references to the execution resources (e.g., tasks do not have assigned execution host) [12]. Here, abstract workflows do not contain references to executable codes for workflow components.

B. Automatic configuration of the workflow

Many different algorithms for the same steps are available in text analytics and handling these algorithms is one of the major ways to support researchers. By making it possible to generate workflows with different combination of algorithms, the comparison of the different results becomes very easy for researchers trying to find out which combinations of algorithms perform best for a given dataset. An example for document classification is to find out which machine learning algorithm performs best for a given dataset. This requires the workflow system to specialize an abstract workflow like the one shown in Figure 1 by systematically searching through all combinations possible and generating executable workflows. It also requires that the workflow system checks whether the combination of algorithms for a given workflow is valid, and reject it if it is not. For example, if a support vector machine (SVM) is used as a modeler, then the classifier must be an SVM classifier.

C. Customize workflows to data and execution environment

With the specialized steps selected during the configuration, the workflow has to be customized further in order to be executed. Depending on the selected dataset for the document classification, the workflow system should select suitable parameters for the algorithms in the workflow. This can be for instance the total number of target labels for a clustering algorithm. Another example is the selection of algorithms and parameters based on the size of a dataset. This information can be very important to estimate the execution time and therefore be relevant for the selected machine learning algorithm. In some cases it might be useful not to execute a workflow or to reduce the dataset in case the number of features exceeds a given value.

Similarly, it is also necessary to customize the workflow for a specific execution environment. Some algorithms can be run on a desktop computer in minutes for a reasonable

sized dataset, others may take many hours of computation. The workflow system therefore must be sensitive to the computation needed by each algorithm and convey to the user the cost of running them with their dataset.

D. Assisting researchers and reducing complexity

A workflow system can offer a unique framework for researchers to experiment with different algorithms and datasets enabling them to focus on the overall performance of the alternative workflows without paying attention to details needed to set up algorithms. The codes are already installed and are part of the workflow system, so there is no need to install new software in order to experiment with a different algorithm. The configuration and the customization of the workflows helps them to focus on their goals, because it automates and manages recurring tasks. Students also benefit from these capabilites tremendously. It is often challenging for students to configure and use complex machine learning algorithms with their individual constrains. This circumstance affects the learning progress negatively and is an additional obstacle for students. Our approach prevents common problems by validating the workflows prior to their execution [13]. Invalid combinations of components or the wrong use of input datasets and parameters abort with an error, so that a student can comprehend errors in early stages.

IV. Implementation

In our work we use the Wings workflow system [6], [8], [7]. Wings is unique in that it uses semantic workflow representations to describe the kinds of data and computational steps in the workflow. Wings can reason about the constraints of the workflow components (steps) and the characteristics of the data and propagate them through the workflow structure. In contrast, most other workflow systems focus either on execution management or on including extensive libraries of analytic tools [4], [14], [15]. Semantic reasoning is used to provide interactive assistance and automation in many aspects of workflow design and configuration. Elsewhere we describe the algorithms used in Wings to specialize abstract workflows and to explore the experiment design space [6], [8]. In [13], we show details of the interaction of a user with Wings through its web-based user interface.

We describe in this section the text analytic workflows that we developed. The software components can have specific requirements on the datasets they are processing. These requirements are expressed in metadata on the datasets. For document classification and clustering, it is necessary to create the datasets and components with their metadata and requirements. Once the components and datasets are created the actual workflows can be designed.

A. Datasets

Text analytics research uses many different datasets to perform experiments and to compare the results along with different algorithms. We selected some very common datasets and use them in the prototype to run experiments. The WebKB [16] dataset is a set of classified web pages. It was collected from computer science departments of various universities. The 20 Newsgroups [17] dataset contains newsgroup documents divided by topic. The Reuters [18] dataset contains manually classified documents from the Reuters newswire. Table I summarizes the datasets with some of their metadata characteristics. Each dataset has a different number of labels for classification and each document can be either multi-labeled or single-labeled. Another property of the dataset is the total number of documents, which is valuable information since it tells us something about the size of the data sets.

TABLE I: datasets with their properties

dataset	# Label	Label	Total # Docs
WebKB	4	Single	4199
20 Newsgroups	20	Single	18821
Reuters R8	8	Single	7674
Reuters R52	52	Single	9100
Reuters R10	10	Multi	7803

All of the datasets have documents in the same data format and are split into a training and a testing subsets. The metadata information from Table I are stored in the Wings Data Catalog and are represented in RDF.

B. Components

Figure 2 shows the most important components from this domain. In addition to these, there are more steps for the preprocessing of the datasets and for handling multi-labeled datasets. The various levels in the figure have different meanings. The *component* node at the top points to abstract components on the second level of the figure. The *classifier* and *modeler* components are shown in the same node. They include Naive Bayes (NB), Gaussian Mixture Models (GMM), k-Nearest Neighbor (kNN), Decision Tree (C4.5) and Support Vector Machines (SVM). The implementations are used from the framework Weka [19]. For the SVM we also use the LIBLINEAR [20] implementation. It achieves similar performance without using kernels for document classification, resulting in much faster overall execution time. Term weighting is one of the pre-processing steps to create a word vector space model of the documents. The *term weighting* component class includes inverse document frequency (IDF), term frequency (TF) and the common term frequency-inverse document frequency (TF-IDF). We implemented these algorithms ourselves both in Java and Python. We used a *stemmer* to handle morphological variation prior to the application of the *term weighting* on the dataset. We used the common Porter stemmer in his original C and a Java version as well as the Lovins stemmer in Java. The feature selection component is used to select the best features by a given score. The score for the feature selection is computed by using one of the *correlation score* components. We have included the chi-square (CS), information gain (IG) and mutual information (MI) [21] algorithms. For document clustering we use implementations from CLUTO [22]. CLUTO is a software package written in C for clustering

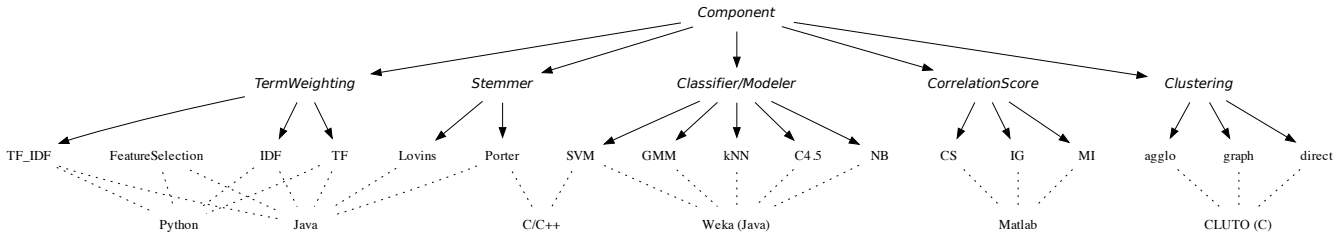


Fig. 2: Hierarchy of abstract component classes and concrete components, indicating their implementation

with various ways to analyze the characteristics of the clusters for low- and high-dimensional data.

Some components have constraints as pre- and postconditions on the datasets they use. The constraints for the datasets are expressed in the Jena rule syntax, which can be found in [23]. Jena is a framework written in Java and can be used as a rule-based inference engine in combination with RDF. We use several types of rules:

- **Validation rules:** A rule that makes sure only single-labeled datasets are used for the correlation score component is shown in Listing 1. It is applied on the correlation score component and selects the input dataset called feature. In case the multi-label property of the dataset is true, the correlation score component is invalidated. This means that no workflows will be generated with that component.

```
(?c rdf:type pcdom:CorrelationScoreClass)
(?c pc:hasInput ?idv)
(?idv pc:hasArgumentID "Feature")
(?idv dcdom:isMultiLabel "true"^^xsd:boolean)
-> (?c ac:isInvalid "true"^^xsd:boolean)
```

Listing 1: Using only single-label data for correlation score

- **Metadata generation rules:** Some rules describe the properties of output datasets of components. The Porter stemmer is a concrete implementation of the abstract stemmer component. In Listing 2 a rule sets a property called *usedStemmer* in the output dataset to denote it was processed through a Porter stemmer. This is often very useful for other components to check which were applied.

```
(?c rdf:type pcdom:PorterStemmerClass)
(?c pc:hasOutput ?odv)
(?odv pc:hasArgumentID "Stem")
-> (?odv dcdom:usedStemmer "PorterStemmer")
```

Listing 2: Setting the used stemmer

- **Execution based component selection:** The workflow can be executed on different execution environments. It can be executed with local resources or with high-performance cyberinfrastructure resources. The current execution environment is compared with the metadata from the classifier component. Components with inappropriate execution environments are rejected during the validation process.

```
(?c rdf:type pcdom:LibSVMClass)
(?c pc:hasInput ?idv)
(?idv pc:hasArgumentID "Model")
(?idv dcdom:usesExecEnvironment "localhost")
-> (?c ac:isInvalid "true"^^xsd:boolean)
```

Listing 3: Selecting component for execution environment

C. Sample Workflow

The components and datasets described in the previous sections are used to create the actual workflows in Wings. Figure 3 illustrates the entire document classification workflow using state-of-the-art methods. The rectangles correspond to components, while the ovals are datasets in green or parameters in orange. The abstract components are marked with dashed boxes. The workflow starts with the training and testing sets at the very beginning. Both datasets go through the same preprocessing steps in the workflow. After stop words and small words (e.g., ”the”, ”of”) are removed, the datasets are stemmed with one of the possible concrete stemmer implementations. The stemmed datasets are used to generate a word vector space model representation using a term weighting component. The last preprocessing step formats the datasets into a sparse data format to reduce the memory usage, because the word vector representation is populated primarily with zeros. A vocabulary with all words appearing in the datasets is needed for this step since an unique identifier has to be assigned for every word. After both training and test set are preprocessed, feature selection is applied on the training set. First, a correlation score is computed for every feature, then the best features are selected based on the individual scores by the parameter *percentage*. The selected features are now used to train a model. The classifier uses the generated model to compute the predictions for the test data that contains parameters for the classifier. The very last component is only necessary to compute accuracy metrics based on the prediction from the classifier. Each workflow also has rules to express constraints:

- **Constraints on data:** An important constraint is that the training and testing data are not equal so the preprocessing components should use the same files:

```
TrainDoc wflow:hasDifferentDataFrom TestDoc
TrainVoc wflow:hasSameDataAs TestVoc
```

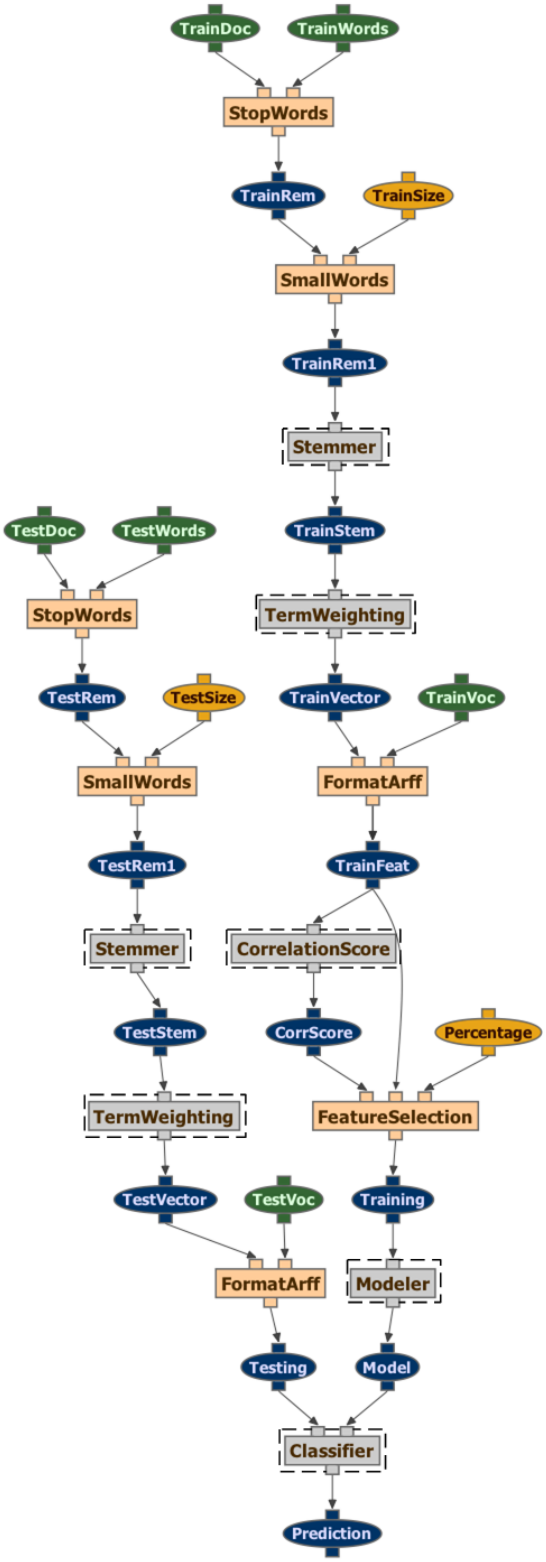


Fig. 3: Document classification workflow in Wings

```
TrainWords wflow:hasSameDataAs TestWords
```

Listing 4: Training and test sets have to be different

- **Constraints on components:** Classifier and modeler always have to use the same machine learning algorithm. As an example one can use a support vector machine to train a model and then use this model with a support vector machine classifier. Invalid combinations are rejected by this rule.

```
(?c rdf:type pcdom:ClassifierClass)
(?c pc:hasInput ?idv)
(?c pc:hasOutput ?odv)
(?idv pc:hasArgumentID "Model")
(?odv pc:hasArgumentID "Prediction")
(?idv dcdom:usedModeler ?s1)
(?odv dcdom:usedClassifier ?s2)
notEqual(?s1,?s2)
-> (?c ac:isInvalid "true"^^xsd:boolean)
```

Listing 5: Use the same algorithm for classifier and modeler steps

Moreover, we have other workflows to do document clustering and generate visualizations. A scientist might use another workflow to visualize the correlation score in order to adjust the settings for a dataset.

V. Results

We have executed the workflow from Figure 3 with different configurations and summarize the results next. A scientist executing experiments is supported by the system in several ways:

- **Fewer workflows generated:** Generating all possible component combinations of the workflow illustrated in Figure 3 would result in some workflow instances that are not executable or produce wrong results. The overall number of possible specialized workflows in this example workflow template is 2,700, which are all the possible combinations of concrete components. In our system, however, only valid combinations of the components are generated. The overall number of generated workflows is already reduced to only 90 specialized workflows, all guaranteed to be valid.
- **Automatic configuration and parameter setup:** Many of the used components in document classification require parameters, that can be elaborated from the metada of the used datasets. A typical example for this is the number of labels in a dataset. This information can be used to set the parameter to indicate the number of target clusters for the clustering components.
- **Generating metadata for workflow results:** The document classification workflow can be executed in many different variations. With all the different implementations, there are 90 different ways to execute the workflow. This complexity of different possibilities to run experiments makes it crucial to understand which algorithms were applied to compute a specific outcome. This is achieved by propagating all the metadata of a dataset through the entire workflow using rules as in Listing 2. After a

component or applied a computation on a dataset, it stores the changes in the metadata properties of the new dataset. As an example the metadata generated for a computed prediction file are summarized in Listing 6.

```
Prediction dcdom:usedTrainFile "WebKB_Train"
Prediction dcdom:usedTestFile "WebKB_Test"
Prediction dcdom:usedStemmer "PorterStemmer"
Prediction dcdom:usedTermWeighting "TF_IDF"
Prediction dcdom:usedModeler "NaiveBayes"
Prediction dcdom:usedClassifier "NaiveBayes"
Prediction dcdom:smallWordsSize 3
Prediction dcdom:selectedFeatures 10
Prediction dcdom:usedCorrelationScore "CS"
Prediction dcdom:isMultiLabel false
```

Listing 6: Metadata of the resulting Prediction

Keeping this provenance information along with the datasets helps to comprehend how results were computed. It also allows users to query across workflow executions and find results based on their metadata (e.g., find all predictions for WebKB dataset or from multi-label data).

- **Heterogenous implementations:** Figure 2 illustrates how heterogenous the different component implementations are. Users see the workflow's abstract steps and do not need to worry about what packages or implementations are used in the end. A classifier might take advantage of additional tools (e.g., Matlab) or use libraries. The framework hides this details from the user and avoids studying manuals and documentation for the different technologies.
- **Method synthesis based on execution environment:** In some situations a researcher might run the document classification workflow on a smaller dataset or a subset to figure out which algorithms perform best for the selected dataset. He might also evaluate different parameter settings to find suitable values for them. As he runs this tests only on a small subset, he is likely to perform well simply on a laptop or workstation. As soon as the same researcher wants to apply his customized workflow on a large scale dataset, he might consider executing the workflows on a high-performance cyberinfrastructure. Switching the execution environment causes the system to select different algorithms and change parameter settings for the components.
- **Simplicity of use of sophisticated methods:** The document classification expertise is incorporated in the design of the prototype workflow. This expertise contains the information about the necessary steps, their correct ordering and the possible algorithms used for every step. Users can now perform experiments with this workflow even when they are not an experts in this domain. The necessary user interactions are thus reduced to the selection of the dataset and the algorithms one wants to use.

Figure 4 illustrates classification results from executing the document classification workflow with two of the datasets and varying term weighting, correlation score, machine learning algorithms and different values for the feature selection.

We used a word list with the most common english words to remove stop words and deleted words with less than 4 characters. The SVM together with chi-square and TF-IDF performed best in our experiments.

VI. Related Work

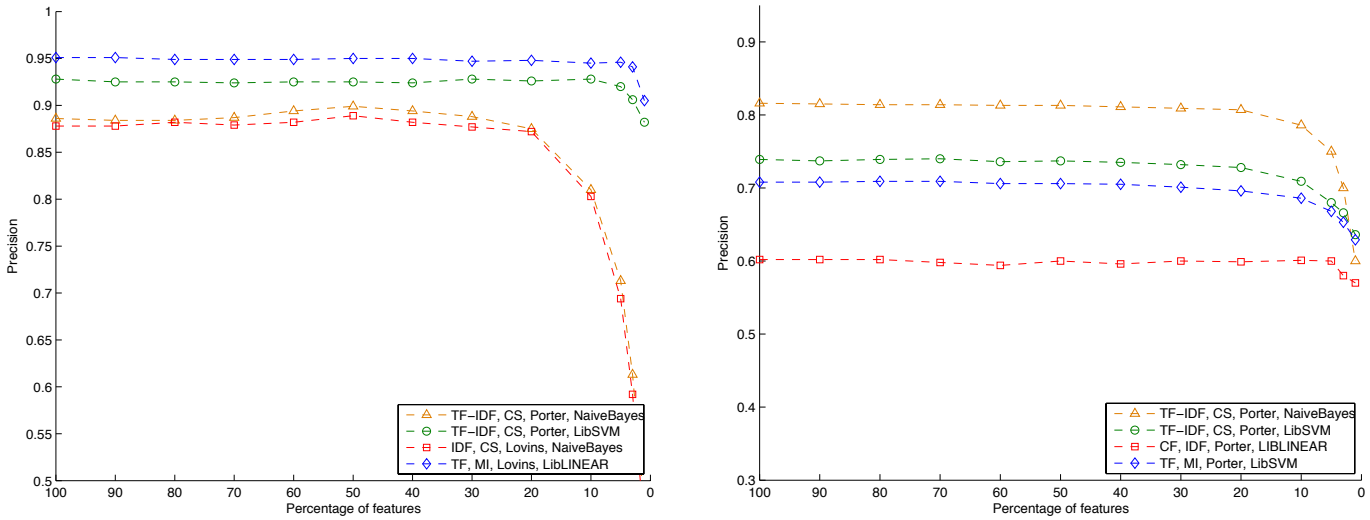
In [24], prominent members of the machine learning community argue for the need to share software and datasets to facilitate experimentation and learning. There are already widely-used libraries such as Weka [19]. The popularity of these systems demonstrates the demand for accessible machine learning codes. Although Weka provides basic functionality to compose codes into workflows, it does not provide any predefined workflows.

Gestalt [9] is a user-centered environment for data analytics that is designed for programmers and guides them through pipelines that include data cleansing and visualization steps. It focuses on classification tasks. A workflow approach for text analytics is used in the IBM UIMA system [25], but it requires manual construction of the workflow including the interfaces between different components. None of these approaches provide a structured framework for exploration or learning.

VII. Conclusion

We have presented a framework for text analytics that uses semantic workflows to assist scientists to efficiently reuse complex end-to-end methods and customize them to their datasets. The framework automatically specializes abstract workflows to systematically explore all valid combinations of algorithms. The framework allows scientists to quickly perform experiments with different methods, understand the relative impact of different data preparation strategies, and to explore the relative merits of different algorithmic choices and their effect in the overall performance. The framework can be easily adapted for other data analytics tasks in many scientific domains.

There are several important benefits to our framework. The overall number of generated experiments is much smaller, because only valid workflows a generated and invalid ones are rejected by the framework. The generated results are enriched with provenance information collected during the workflow execution. This is especially useful to comprehend the results after many experiments with different components were executed. The system also handles heterogenous implementations depending on the current execution environment. A user might switch the execution environment to run computations on a high-performance cyberinfrastructure without having to change the workflow. All these facilities reduce the necessary user interactions and in the end hide complexity involved in the execution and configuration of the experiments. We believe this framework can have a broader impact as a learning environment for novice researchers, students, and scientists in other areas of data analytics.



(a) Reuters R8

(b) 20 Newsgroups

Fig. 4: Classification results for two of the datasets

ACKNOWLEDGMENTS

This research arouse in association with the Elite Graduate Program Software Engineering of the University of Augsburg, Technical University Munich and the Ludwig-Maximilians-University Munich and was funded in part by the National Science Foundation under award CCF-0725332.

REFERENCES

[1] I. J. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag, 2006.

[2] Y. Gil, "From Data to Knowledge to Dicoveries: Scientific Workflows and Artificial Intelligence," *Scientific Programming*, vol. 17, 2009.

[3] D. De Roure, C. Goble, and R. Stevens, "The design and realisation of the Virtual Research Environment for social sharing of workflows," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 561–567, 2009.

[4] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: Visualization meets Data Management," in *ACM SIGMOD*, 2006, pp. 745–747.

[5] C. Wroe, C. A. Goble, A. Goderis, P. W. Lord, S. Miles, J. Papay, P. Alper, and L. Moreau, "Recycling workflows and services through discovery and reuse," *Concurrency and Computation: Practice and Experience*, vol. 19, pp. 181–194, 2007.

[6] Y. Gil, V. Ratnakar, J. Kim, P. A. González-Calero, P. Groth, J. Moody, and E. Deelman, "WINGS: Intelligent Workflow-Based Design of Computational Experiments," in *IEEE Intelligent Systems (IS'11)*, London, UK, 2011.

[7] Y. Gil, P. Groth, V. Ratnakar, and C. Fritz, "Expressive Reusable Workflow Templates," in *Proceedings of the Fith IEEE International Conference on e-Science*, Oxford, UK, 2009.

[8] Y. Gil, P. González-Calero, J. Moody, and V. Ratnakar, "A Semantic Framework for Automatic Generation of Computational Workflows Using Distributed Data and Component Catalogs," *To appear in the Journal of Experimental and Theoretical Artificial Intelligence*, 2011.

[9] K. Patel, N. Bancroft, S. Drucker, J. Fogarty, A. Ko, and J. A. Landay, "Gestalt: Integrated Support for Implementation and Analysis in Machine Learning Processes," in *Proceedings of the ACM Symposium on User Interface Software and Technology UIST*, 2010.

[10] M. Hall, Y. Gil, and R. Lucas, "Self-Configuring Applications for Heterogeneous Systems: Program Composition and Optimization Using Cognitive Techniques," in *Proceedings of the IEEE, Special Issue on Cutting-Edge Computing: Using New Commodity Architectures*, vol. 96, no. 5, 2008.

[11] A. W. Brown, *Large Scale Component Based Development*. Prentice Hall PTR; 1st edition, 2000.

[12] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda, "Mapping Abstract Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, no. 1, 2003. [Online]. Available: papers/deelman-etal-jogc03.pdf

[13] Y. Gil, V. Ratnakar, and C. Fritz, "Assisting Scientists with Complex Data Analysis Tasks through Semantic Workflows," in *In Proceedings of the AAAI Fall Symposium on Proactive Assistant Agents*, Arlington, VA, Nov. 2010.

[14] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, iss. 10, pp. 1067–1100, 2006.

[15] M. Reich, T. Liefeld, J. Gould, J. Lerner, and J. P. Mesirov, "GenePattern 2.0," *Nature Genetics 38*, vol. 5, doi:10.1038/ng0506-500, pp. 500–501, 2006.

[16] (1998, Jan.) The 4 Universities Data Set. Carnegie Mellon University. [Online]. Available: http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

[17] (2008, Jan.) The 20 Newsgroups data set. [Online]. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/

[18] (2004, May) Reuters-21578. Carnegie Group, Inc. and Reuters, Ltd. [Online]. Available: http://www.daviddlewis.com/resources/testcollections/reuters21578/

[19] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," *ICONIP/ANZIIS/ANNES*, pp. 192–196, 1999.

[20] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, Dec. 2008.

[21] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," in *International Conference on Machine Learning (IDCML'97)*. Morgan Kaufmann Publishers, 1997, pp. 412–420.

[22] G. Karypis. (2006, Oct.) CLUTO - Software for Clustering High-Dimensional Datasets. Karypis Lab, University of Minnesota. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/views/cluto

[23] (2011, Jun.) Jena: Semantic web framework. HP-Lab. [Online]. Available: http://jena.sourceforge.net/documentation.html

[24] S. Sonnenburg, S. Bengio, L. Bottou, Y. LeCun, and K.-R. Müller, "The Need for Open Source Software in Machine Learning," *Journal of Machine Learning Research*, 2007.

[25] D. Ferrucci and A. Lally, "UIMA: An architectural approach to unstructured information processing in the corporate research environment," *Nat. Lang. Eng.*, vol. 10, pp. 327–348, 2004.