CSE Framework: A UIMA-based Distributed System for Configuration Space Exploration

Elmer Garduno[1], Zi Yang[2], Avner Maiberg[2], Collin McCormack[3], Yan Fang[4], and Eric Nyberg[2]

[1] Sinnia, elmerg@sinnia.com
[2] Carnegie Mellon University, {ziy, amaiberg, ehn}@cs.cmu.edu
[3] Boeing Company, collin.w.mccormack@boeing.com
[4] Oracle Corporation, yan.fang@oracle.com

Abstract. To efficiently build data analysis and knowledge discovery pipelines, researchers and developers tend to leverage available services and existing components by plugging them into different phases of the pipelines, and then spend hours to days seeking the right components and configurations that optimize the system performance. In this paper, we introduce the CSE framework , a distributed system for a parallel experimentation test bed based on UIMA and uimaFIT, which is general and flexible to configure and powerful enough to sift through thousands option combinations to determine which represent the best system configuration.

1 Introduction

To efficiently build data analysis and knowledge discovery "pipelines", researchers and developers tend to leverage available services and existing components by plugging them into different phases of the pipelines [1], and then spend hours, seeking for the components and configurations that optimize the system performance. The Unstructured Information Management Architecture (UIMA) [3] provides a general framework for defining common types in the information system (*type system*), designing pipeline phases (*CPE descriptor*), and further configuring the components (*AE descriptor*) without changing the component logic. However there is no easy way to configure and execute a large set of combinations without repeated executions, while evaluating the performance of each component and configuration.

To fully leverage existing components, it must be possible to automatically explore the space of system configurations and determine the optimal combination of tools and parameter settings for a new task. We refer to this problem as *configuration space exploration*, which can be formally defined as a constraint optimization problem. A particular information processing task is defined by a configuration space, which consists of m_t components that define each of the n phases with corresponding configurations. Given a limited total resource capacity $\mathcal{C}$ and input set $\mathcal{S}$, **configuration space exploration** (CSE) aims to find the trace (a combination of configured components) within the space that achieves the highest expected performance without exceeding $\mathcal{C}$ total cost. Details on the mathematical definition and proposed greedy solutions can be found in [6].

In this paper, we introduce the CSE framework implementation, a distributed system for parallel experimentation test bed based on UIMA and uimaFIT [4]. In addition, we highlight the results from two case studies where we applied the CSE framework to the task of building biomedical question answering systems.

```yaml
# main.yaml
collection-reader:
  inherit: input-reader
  dataset: TRECGEN06
  file: /trecgen06.txt

pipeline:
  - inherit: ecd.phase
    name: tokenization
    options:
     - inherit: stanford-nlp
     - inherit: linepipe

  - inherit: ecd.phase
    name: s/acynonym-exp
    options:
     - inherit: mesh
     - inherit: entrez-gene
     - inherit: umls

  - inherit: ecd.phase
    name: retrieval
    options:
     - inherit: indri
     - inherit: lucene

  - inherit: ecd.phase
    name: reranking
    options:
     - inherit: proximity
     - inherit: base.noop

  - inherit: prec-evaluator

post-process:
  - inherit: map-evaluator
  - inherit: report-gen
    builders:
      - inherit: map-report
```

```yaml
# indri.yaml
class: IndriWrapper
cross-opts:
  smoothing: [0.1, 0.3, 0.5, 0.7]
  weighting: [0.1, 0.3, 0.5, 0.7]
```

Fig. 1. Example YAML-based pipeline descriptors

2 Framework Architecture

We highlight some features of the implementation in this section. Source code, examples, documentation, and other resources are publicly available on GitHub[5]. To benefit developers who are already familiar with UIMA framework, we have developed a CSE tutorial in alignment with the examples in the official UIMA tutorial.

Declarative descriptors. To leverage the CSE framework, users need to specify how the components should be organized in the pipeline, which values need to be specified for each component configuration, which is the input set, and what measurement metrics should to be applied. Analogous to a typical UIMA CPE descriptor, components, configurations, and collection readers in the CSE framework are declared in *extended configuration descriptors* which are based on the YAML format. An example of the main pipeline descriptor and a component descriptor are shown in Figure 1.

Architecture. Each pipeline can contain an arbitrary number of AnalysisEngines declared by using the *class* keyword or by *inherit*ing configuration options from other components by name. Combinations of components are configured using an *options* block and parameter combinations within a component are configured on a *cross-opts* block. To take full advantage of the CSE framework capabilities, users inherit from a *cse.phase*, a CAS multiplier that provides, option multiplexing, intermediate resource persistence and resource management for long running components. The architecture also supports grouping options into sub-pipelines as a convenient way of reducing the configuration space for combinations whose performance is already known.

Evaluation. Unlike a traditional scientific workflow management system, CSE emphasizes the evaluation of component performance, based on user-specified evaluation metrics and gold-standard outputs at each phase. In addition the framework keeps track of the performance of all the executed traces, this allows inter-component evaluation and automatic tracking of performance improvements through time.

Automatic data persistence. To support further error analysis and reproduction of experimental results, intermediate data (CASes) and evaluation results are kept in a repository accessible from any trace at any point during the experiment. To prevent duplicate execution of traces the system keeps track of all the execution traces an recovers those CASes whose predecessors have

[5] http://oaqa.github.io/

Table 1. Case study result

	# Comp	# Conf	# Trace	# Exec	DocMAP			PsgMAP		
					Max	Median	Min	Max	Median	Min
Participants	∼1,000	∼1,000	92	∼1,000	.5439	.3083	.0198	.1486	.0345	.0007
CSE	13	32	2700	190,680	.5648	.4770	.1087	.1773	.1603	.0311

already been executed. Also the overall results from experiments are kept in a historical database to allow researchers to keep track of the performance improvements along time.

Configurable selection and pruning. If gold-standard data is provided for a certain phase, then components up to that phase can be evaluated. Given the measured cost of executing the components provided, components can be ranked, selected or pruned for evaluation and optimization of subsequent phases. The component ranking strategy can be configured by the user; several heuristic strategies are implemented in the open source software.

Distributed architecture. We have extended the CSE framework implementation to execute the task set in parallel on a distributed system using JMS. The components and configurations are deployed into the cluster beforehand. The execution, fault tolerance and bookkeeping are managed by a master server. In addition we leverage the UIMA-AS capabilities to execute specific configurations in parallel as separate services directly from the pipeline.

3 Building biomedical QA Systems via CSE

As a case study, we apply the CSE framework to the problem of building effective biomedical question answering (BioQA) on two different tasks.

In one case, we employ the topic set and benchmarks, including gold-standard answers and evaluation metrics, from the question answering task of the TREC Genomics Track 2006, as well as commonly-used tools, resources, and algorithms cited by participants. The implemented components, benchmarks, task-specific evaluation methods are included in domain-specific layer named BioQA, which was plugged into the BaseQA framework.

The configuration space was explored with the CSE framework, automatically yielding an optimal configuration of the given components which outperformed published results for the same task. We compare the settings and results for the experiment with the official TREC 2006 Genomics test results for the participating systems in Table 1. We can see that the best system derived automatically by the proposed CSE framework can outperform the best participating system in terms of both DocMAP and PsgMAP, with fewer, more basic components. This experiments ran on a 40 node cluster during 24 hours allowing the execution on 200K components over 2,700 execution traces. More detailed analysis can be found in [6].

We also used the CSE framework to automatically configure a different type of biomedical question answering system for the QA4MRE (Question Answering for Machine Reading Evaluation) task at CLEF. The CSE framework identified a better combination, which achieved 59.6% performance gain over the original pipeline. Details can be found in the working note paper [5].

4 Related Work

Previous work has been done on this area, in particular DKPro Lab [2] a flexible lightweight framework for parameter sweep experiments and the U-Compare [7] framework, an evaluation

platform for running tools on text targets and compare components, that generates statistics and instance-based visualizations of outputs.

One of the main advantages of the CSE framework is that it allows the exploration of very large configuration spaces by distributing the experiments over a cluster of workers and collecting the statistics on a centralized way. Another advantage on the CSE framework is that configurations can have arbitrary nesting levels as long as they form a DAG by using sub-pipelines. Also results can be compared end-to-end at a global level to understand overall performance trends on time. One area where CSE could take advantage of the aforementioned frameworks is on having a graphical UI for pipeline configuration, better visualization tools for combinatorial and instance-based comparison and a more expressive language for workflow definition.

5 Conclusion & Future Work

In this paper, we present a UIMA-based distributed system to solve a common problem in rapid domain adaptation, referred to as Configuration Space Exploration. It features declarative descriptors, evaluations, automatic data persistence, global resource caching, configurable configuration selection and pruning, and distributed architecture. As a case study, we applied the CSE framework to build a biomedical question answering system, which incorporated the benchmark from TREC Genomics QA task, and the results showed the effectiveness of the CSE framework system.

We are planning to adapt the system to a wide variety of interesting information processing problems to facilitate rapid domain adaption and system building and evaluation of the community. For educational purpose, we are also interested in adopt the CSE framework as an experiment platform to teach students the principled ways to design, implement and evaluate an information system.

Acknowledgement. We thanks Leonid Boystov, Di Wang, Jack Montgomery, Alkesh Patel, Rui Liu, Ana Cristina Mendes, Kartik Mandaville, Tom Vu, Naoki Orii, and Eric Riebling for the contribution to the design and development of the system and valued suggestions to the paper.

References

1. D. Ferrucci et al. Towards the Open Advancement of Question Answering Systems. Technical report, IBM Research, Armonk, New York, 2009.
2. R. E. de Castilho and I. Gurevych. A lightweight framework for reproducible parameter sweeping in information retrieval. In *Proceedings of the DESIRE'11 workshop*, New York, NY, USA, Oct. 2011.
3. D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4), Sept. 2004.
4. P. Ogren and S. Bethard. Building test suites for UIMA components. In *Proceedings of the (SETQA-NLP 2009) workshop*, Boulder, Colorado, June 2009. Association for Computational Linguistics.
5. A. Patel, Z. Yang, E. Nyberg, and T. Mitamura. Building an optimal QA system automatically using configuration space exploration for QA4MRE'13 tasks. In *Proceedings of CLEF 2013*, 2013.
6. Z. Yang, E. Garduno, Y. Fang, A. Maiberg, C. McCormack, and E. Nyberg. Building optimal information systems automatically: Configuration space exploration for biomedical information systems. In *Proceedings of the CIKM'13*, 2013.
7. K. Yoshinobu, W. Baumgartner, L. McCrohon, S. Ananiadou, K. Cohen, L. Hunter, and J. Tsujii. U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, 25(15):1997–1998, 2009. in press.