AFRL-RI-RS-TR-2018-218

D-CUBE: DELITEFUL DEEPDIVE FOR DOMAIN-SPECIFIC INDEXING AND SEARCH FOR THE WEB

LELAND STANFORD JUNIOR UNIVERSITY

SEPTEMBER 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2018-218 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
PETER ROCCI
Work Unit Manager

/ S /
TIMOTHY A. FARRELL
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
SEPTEMBER 2018	FINAL TECHNICAL REPORT	SEP 2014 – MAR 2018

4. TITLE AND SUBTITLE

D-CUBE: DELITEFUL DEEPDIVE FOR DOMAIN-SPECIFIC INDEXING AND SEARCH FOR THE WEB

5a. CONTRACT NUMBER
FA8750-14-2-0240

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Oyekunle Olukotun

5d. PROJECT NUMBER
MEMX

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
09

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Board of Trustees of the Leland Stanford Junior University
3160 Porter Dr., Ste. 100
Palo Alto, CA, 94304

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIED
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-218

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes the three main components of our team's work on the MEMEX program. The first component, called Delite, focuses on creating a higher-level domain specific languages (DSLs) that automate many of the tedious portions of acquiring, extracting, and analyzing search data. The second component is the DeepDive Knowledge Base Construction Engine which has enabled users to efficiently achieve enhanced domain-specific extraction performance. The third component, called Snorkel, simplifies the DeepDive pipeline to enabling data analysts to focus on the task at hand. Delite, DeepDive and Snorkel are widely deployed by both industry and academia to support critical applications and research.

15. SUBJECT TERMS

Domain Specific Languages, Knowledge Base Construction, Machine Learning, Multicore, FPGA

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**PETER ROCCI**
U	U	U	UU	25	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

1.0 SUMMARY

The overall goal of the MEMEX project was to develop software systems that would enable analysts to quickly build high-quality, scalable domain-specific indexing and search engines. This report describes the main components of our work on this project, which in concert focus on developing computational techniques and software tools to improve analyst performance in the context of the imprecision, ambiguity, and the sheer variety of language and structure on the web.

This report describes the three main components of our team's work on the MEMEX program. The first component focuses on creating a higher-level domain specific language (DSL) that automates many of the tedious portions of acquiring, extracting, and analyzing search data. We have demonstrated how the Delite framework can be used to make search and acquisition first-class citizens in the DSL via integration of a high-throughput backend. Delite is a framework for building compilers for high-performance Domain Specific Languages that can be used to target heterogeneous architectures (multicore, GPU, cluster, FPGA) [30]. Integration of this work and a variety of other performance improvements with the second component of our system, the DeepDive Knowledge Base Construction Engine [10], has enabled users to efficiently achieve enhanced domain-specific extraction performance using a variety of hardware architectures[. In the context of human trafficking, DeepDive running on a Delite backend has provided scalable, domain-specific extraction capability for multiple users associated with the program. Building on lessons learned from this process, the Snorkel software package [4] was developed as the third component of our work in order to

abstract away from the user tasks in the DeepDive pipeline that required substantial machine learning knowledge, enabling analysts to focus on the task at hand. Like DeepDive before it, Snorkel and its associated software projects [1, 2, 5, 8, 12] are widely deployed by both industry and academia to support critical applications and research.

2.0 INTRODUCTION

The overall goal of the MEMEX project was to develop software systems that would enable analysts to quickly build high-quality, scalable domain-specific indexing and search engines. In this context, our main technical objectives were to make domain-specific search dramatically more cost effective and more usable by a larger set of analysts than is possible today.

Our experience indicates that creating a high-quality domain-specific search and extraction system is not a one-shot task, but is a dialogue. In this dialogue, an analyst repeatedly cycles through three phases: improving the extraction systems, acquiring new data, and refining (possibly statistical) analysis. To facilitate this dialogue, we have worked to improve each of the three phases: (1) acquisition and search by making each first class citizens in our language, (2) extraction by making inference procedures faster, easier, and more automatic, and (3) analysis by making the analysis and diagnosis cycle interactive.

- Problem 1: A higher-level language for search tasks. Many routine operations in current search and analysis frameworks such as DeepDive [10] can be painful and slow, which forces users to copy and paste code from examples or to use suboptimal features. These are clear signals we do not have the right abstractions to make this framework analyst friendly. Our solution: Delite [30] allows one to specify tight, domain-specific abstractions and creates a Read-Eval-print Loop

(REPL) instance for each DSL, which allows for an interactive method for users to construct and debug their search indexes. We envision that as they are performing their downstream analysis, they will often need to dig back into the source extraction material, perhaps even improving portions of the extraction for their needs. We proposed to use Delite to construct a higher-level domain specific language that automates many of the tedious portions of acquiring, extracting, and analyzing search data. This involves (a) making search and acquisition first-class citizens in the DSL, which requires that we extend Delite to support a high-throughput backend, (b) extending DeepDive's extraction capability and integrating it with Delite, and (c) extending the analysis and diagnosis capability of both frameworks.

- Problem 2: Reducing analyst time through a high performance infrastructure. A higher performance infrastructure is critical to enable more automation in that infrastructure. Our Solution: We proposed to improve the scalability and performance of DeepDive by porting it to Delite, which allows us to take advantage of graphics Processing Units (GPUs), Non-Uniform Memory Access (NUMA) machines, and clusters of machines to achieve dramatically higher levels of performance. This process involved several crucial steps. First, we created a high-throughput backend, where a key tradeoff space is how different models of isolation and fault tolerance affect throughput. Next, we used this speed to automate many of the testing, inference, and learning procedures that DeepDive originally forced on the analyst. In addition to accelerating DeepDive

via integration with such improvements as a fast Gibbs sampling architecture [20], we investigated a number of other techniques to enable high-performance infrastructure that can reliably support the analyst. These include development of a reconfigurable hardware architecture to support parallel execution [7], design of a deep learning performance benchmark [6], construction of a distributed Gibbs sampler [21], and eventually transitioning away from Gibbs sampling to a matrix approximation algorithm [1] that can be used to solve many of the same underlying problems for which our systems were developed with both increased speed and more favorable mathematical properties.

- Problem 3: Accelerating the Analysis Cycle with Software 2.0. The systems proposed for this work such as Delite and DeepDive were research prototypes, not industrial- strength software. We planned to transition our final systems to a broader audience, which required that we take care of a host of mundane but important details: reliable execution, informative error messages, documentation, copious examples and frequently asked questions, an ability to work in multiple configurations. Our Solution: We originally proposed to have the two software developers begin hardening the existing code base and then in later stages of the proposed work, they would work on transitioning the new features to practice. In fact, DeepDive was commercialized as Lattice, with a full software engineering team having been built around the product. Further, our development of the Snorkel system [4] as a follow on to DeepDive has been aimed at leveraging emerging techniques at the intersection of machine learning and software

engineering (“Software 2.0”) to allow for non-expert users to leverage DeepDive-like knowledge base construction capability.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 A Higher-level Language for Search Tasks

Enabling analysts to perform both search and analysis on domain-specific tasks at scale generally requires large amounts of processing power. It is currently a significant burden to develop the best implementations of data analysis algorithms for all the varieties of parallel architecture (multicore, GPU, cluster, FPGA). The traditional library-based approach to this problem has portability and versatility limitations. Instead, we have pursued an approach to integrating data-analysis applications such as DeepDive with a domain specific language. We have leveraged a DSL development framework called Delite to simplify the process of developing high-performance easy to use DSLs. The components of the Delite framework are shown in Figure 1.

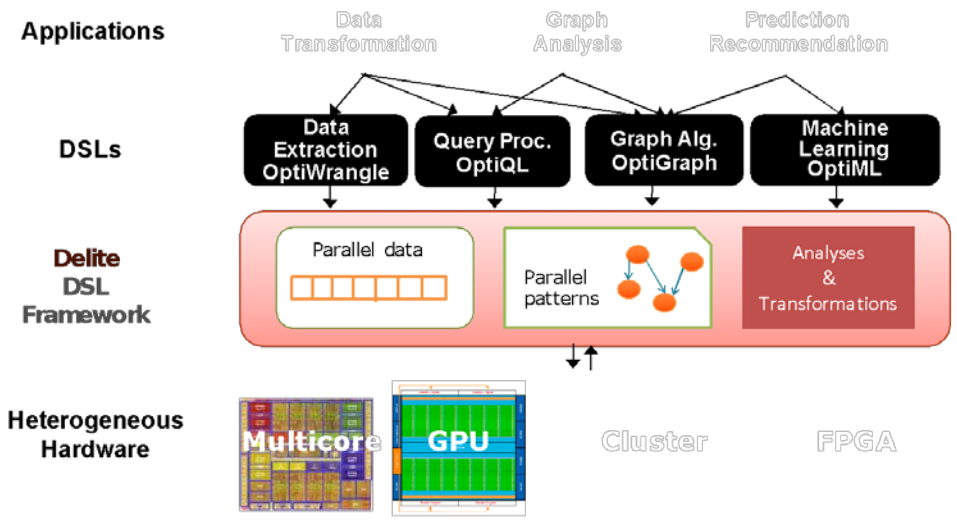

Figure 1. The Delite DSL Framework.

Delite substantially reduces the burden of developing high-performance DSLs by providing common reusable components that enable DSLs to quickly and efficiently target heterogeneous hardware [30].

3.2 Reducing Analyst Time with High Performance Infrastructure

In order to reduce analyst time on critical tasks, we have focused on a variety of strategies to improve underlying system performance. As a first step, we leveraged a DSL to create an efficient Gibbs sampler to provide substantial (over 50x, in some cases) speedup for running analysis systems like DeepDive. As a followup to this work, we developed a distributed Gibbs sampling architecture that enabled computations using factor graphs that could not be stored in memory on a single machine. Further, over the

course of the MEMEX program, the commoditization of deep learning models and the importance of providing appropriate supervision became a clear bottleneck for analysts in practice, and we therefore transitioned much of our work to this setting. Specifically, we reformulated the graphical modeling underlying DeepDive into a matrix completion algorithm [1] that has no dependency on the number of unlabeled examples used to learn the accuracies of different rules that contribute to knowledge base creation. This recent advance forms the basis of the next generation of analytics software that our group is providing in the open source [1, 2, 4, 5, 8, 12]. In the DAWNBench project described in [7], we attempted to better understand the different ways in which modern deep learning techniques and software frameworks interact with systems capabilities in a robust fashion.

3.3 Accelerating the Analysis Cycle with Software 2.0

Over the course of the last several years, the Snorkel software system [4] has grown directly out of the DeepDive project, but with an emphasis on a *Software 2.0, weak supervision-first* perspective. In DeepDive, users could specify multiple "tasks" that were jointly inferred over to make final predictions, but doing this required large amounts of time and technical machine learning expertise. For *each* separate task, users would have to specify (i) a separate set of *features* painstakingly encoded by hand, and (ii) manually generated training data, either hand-labeled or created with carefully combined *distant supervision rules.* In Snorkel, the features are learned automatically by multi-task models, which effectively pool samples across tasks to learn better features [1]. Instead,

as shown in Figure 2, users only need to provide labeling functions, via an accessible Jupyter notebook interface, which are then automatically combined and jointly modeled across tasks by Snorkel. The combination of these labeling functions enables automatic generation of probabilistic training labels even for *unlabeled* examples, which can be used to train modern deep learning algorithms at unprecedented scale. Key theoretical advancements underlying Snorkel, when translated to practice, have resulted in a system that allows for domain experts to rapidly, scalably, and reliably leverage machine learning systems for mission-critical tasks across a variety of contexts including intelligence, business, medicine, and basic science.

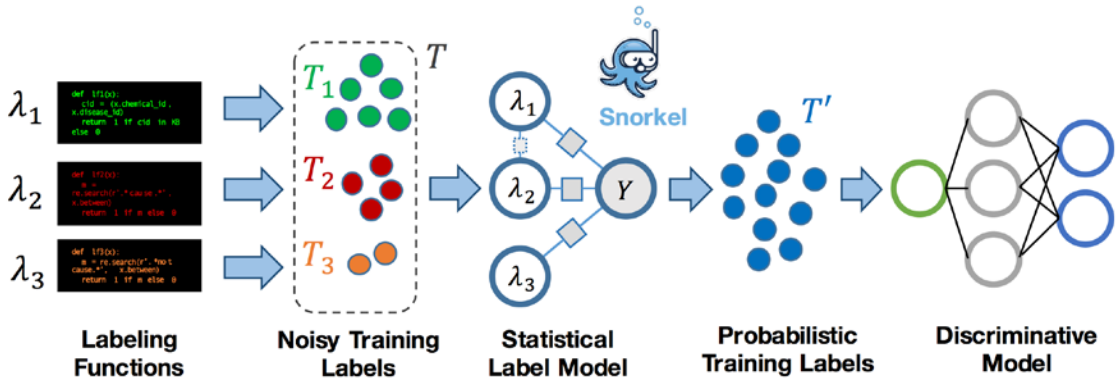

Figure 2. The Snorkel Analysis Pipeline.

4.0 RESULTS AND DISCUSSION

4.1 A Higher-level Language for Search Tasks

Under MEMEX, we successfully integrated DeepDive with a Delite-generated DSL and were able to achieve substantial performance gains in practice. In particular, a Gibbs sampler composed using the Distributed Multi Loop Language (DMLL) demonstrated substantial speedups (in some cases over 50x) over running the naive sequential DimWitted implementation [21]. Today, these techniques form the core of the DeepDive Knowledge Construction System that has been deployed widely in both academic and industry [10].

4.2 Reducing Analyst Time with High Performance Infrastructure

In addition to the DeepDive-Delite integration, under MEMEX our work yielded a number of advances that enabled us to reliably improve the performance of our data analytics stack. These are described below.

Numbskull: Multi-threading Gibbs Sampling

Following up on our work on DMLL, we created a new Gibbs Sampler called Numbskull to power our analytics applications in a scalable fashion. Numbskull consumes substantially less memory than previous engines, making it possible to extract from larger corpora on smaller machine resources. Numbskull started as a Python-based

implementation of Dimmwitted, i.e., as a single-box multi-threaded sampler. Multithreaded Numbskull is within a factor of 2x slower than Dimmwitted but consumes 25% less memory. We also introduced an across- boxes distributable version of Numbskull. The motivation behind distributed Numbskull stems from applications such as MEMEX where the full factor graph cannot be materialized within the memory of a single machine. In distributed Numbskull the factor graph is never materialized on a single machine. This is in contrast to previous techniques that rely on analyzing the fully materialized factor graph before partitioning (either using connected components or finding weakly connected subgraphs). To achieve the above we introduced a partitioning paradigm which we refer to as semantic partitioning. We only require access to the template-specification that generates the underlying factor graph and by employing type analysis we have a new way of distributing the factor graph generated by that template across machines. We follow a master-minions architecture where a central master box is coordinating multiple minion boxes. Our partitioning and sampling approach comes with several guarantees on the mixing-time of Gibbs.

Snorkel MeTaL: Using Matrix Completion to Support Multitask Learning

To improve both user experience and task performance, we applied lessons learned from DeepDive to create Snorkel [4], a new system focused on leveraging user domain knowledge to create large, weakly labeled training sets that can be used to train powerful machine learning models without requiring substantial machine learning expertise on the part of the user. On the technical front, we have developed a new approach to training the underlying generative model, which learns the accuracies of the user-provided heuristics, or "labeling functions," using a masked matrix approximation-style approach [1]. In addition to being simpler and more robust than Gibbs Sampling, this approach does not scale at all with the amount of unlabeled data provided. That is, users can now add in as much unlabeled data as is available to them, increasing end quality without any slowdown in generative model training (aside from the negligible cost of one initial matrix multiply). In addition, this new formulation of our model has led to new, tighter theoretical bounds on the scaling of performance with more unlabeled data.

One of our major pushes has been towards making a new version of Snorkel, Snorkel MeTaL, which handles weak supervision for massively multi-task models that take in many different types of training data, and use it to jointly train models that perform many different *tasks*. For example, rather than using Snorkel ten different times to train ten different extraction models for ten different but related data types, in Snorkel MeTaL, users can train one multi-task model to jointly learn all ten extractors, for improved overall performance (see technical report for experiments on fine-grained relation extraction). And in contrast to current work on multi-task learning, which generally

relies on several existing hand-labeled datasets, Snorkel MeTaL accepts multi-task labeling functions as input---enabling users to specify arbitrarily expansive sets of custom sub-tasks. We are continuing to develop Snorkel MeTaL and plan to merge it increasingly with the current Snorkel framework over the coming months, adding massively multi-task capacity by default.

DAWNBench: Robust Evaluation of Deep Learning Performance

To provide an objective means of quantifying performance of the end-to-end deep learning models that power many modern analytics workloads, we have developed DAWNBench [6], an open benchmark and competition for end-to-end deep learning training and inference. Instead of simply measuring time per iteration (or throughput), DAWNBench measures end-to-end performance in training (e.g., time, cost) and inference (e.g., latency, cost) at a specified state-of-the-art level of accuracy. This provides an objective means of normalizing across differences in computation frameworks, hardware, optimization algorithms, hyperparameter settings, and other factors that affect real-world performance. Our initial release of DAWNBench provides end-to-end learning and inferencetasks including image classification on CIFAR10 and ImageNet, and question answering on SQuAD, along with reference implementations for each task.

Plasticine: Integrating Parallelism into Reconfigurable Hardware

We have developed a new reconfigurable architecture called Plasticine [7]. Plasticine is designed to efficiently execute applications composed of parallel patterns. Parallel

patterns have emerged from recent research on parallel programming as powerful, high-level abstractions that can elegantly capture data locality, memory access patterns, and parallelism across a wide range of dense and sparse data analysis applications.

4.3 Accelerating the Analysis Cycle with Software 2.0

Snorkel: A Software Framework for Weak Supervision

Snorkel [4] continues to be our main software platform for exploring how end-to-end machine learning---or software 2.0---systems can be built by subject matter experts without machine learning expertise, by having them provide weak supervision in the form of noisy labeling functions that programmatically label data. We have continued to build and improve the open source Snorkel project based on feedback from a growing set of collaborators and users, including several teams at Google, Intel, Facebook, BASF, Alibaba, various labs at the Stanford Medical School [27], Stanford VA hospital, and many other users. Software improvements include supporting PyTorch and Tensorflow bindings, making pip installable, and other system and usability improvements. Snorkel now has over 1,200 stars on github, and we are excited about the growing user community around it; to this end, we have continued community outreach and education activities, including most recently an ACM summer school workshop.

Fonduer: Leveraging Weak Supervision over Richly Formatted Data

Traditionally, KBC solutions such as DeepDive and Snorkel have focused on relation extraction from unstructured text or semi-structured data. These systems already support

a broad range of downstream applications such as information retrieval, question answering, medical diagnosis, and data visualization. However, troves of information remain untapped in richly formatted data, where relations and attributes are expressed via combinations of textual, structural, tabular, and visual modalities. In these scenarios, the semantics of the data are significantly affected by the organization and layout of the document. Examples of richly formatted data include webpages, business reports, product specifications, and scientific literature.

In Fonduer [5], we introduce a programming model in which no development cycles are spent on feature engineering. Users only need to specify candidates, the potential entries in the target knowledge base (KB), and provide lightweight supervision rules which capture a user's domain knowledge to programmatically label subsets of candidates, which are used to train Fonduer's deep-learning model. Our user study to evaluate Fonduer's programming model shows that when working with richly formatted data, users rely on semantics from multiple modalities of the data, including both structural and textual information in the document. We have evaluated Fonduer in four real-world applications of richly formatted information extraction, including the MEMEX extraction task, and show that Fonduer enables users to build high-quality KBs that achieve an average improvement of 41 F1 points over state-of-the-art systems.

Coral: Handling Complex Dependency Structures in Supervision

We have also attempted to better capture data comprising a variety of modalities through our Coral project [8], which focuses on weak supervision with image and video data.

Given the difficulty of defining labeling functions over raw pixels, Coral introduces the concept of *domain-specific primitives*, or interpretable characteristics of image and video data that users can exploit to write these labeling functions. Examples of these domain-specific primitives include bounding box information like location, size, and label in natural images, which can be acquired using state-of-the-art object detection algorithms. For medical applications, these include characteristics like the area, perimeter or more complex shape and texture based features for the region of interest in a radiology image.

Given these domain-specific primitives and labeling functions written over them, we can now utilize the same weak supervision model as Snorkel to assign noisy label to image and video training data. Moreover, we can also exploit the structure of these labeling functions that rely on the domain-specific primitives to augment the statistical model that assigns training labels. Specifically, we look at *how the labeling functions are designed* in terms of the inputs they rely on and encode this into our statistical model. This allows Coral to boost the accuracy of the training labels, by up to 3.81 F1 points for a video classification task. Coral also allows users to label previously *unlabeled data*, outperforming fully supervised models with a smaller amount of ground truth label by up to 3.07% accuracy. .

Babble Labble: Deriving Supervision Automatically from Natural Language

With Snorkel, labeling massive training sets is made easier via labeling functions. In the Babble Labble project [2], we explored if we could lower the barrier to entry even further by allowing the user to label training sets using only natural language. While viewing unlabeled examples, the user provides natural language explanations for why examples should be labeled a certain way. These explanations are then parsed into executable labeling functions that can be applied within the Snorkel framework. Surprisingly, we found by applying a small number of common-sense filters to the potential parses generated by the semantic parser, a naive rule-based parser performs nearly as well as a theoretical perfect parser. In other words, we can build a parser once and use it in many domains without additional training, making it easy to collect natural language supervision in new domains with little overhead. We evaluated this approach on three relation extraction tasks, finding that on average Babble Labble requires 5-100x fewer user inputs (explanations) than a traditional supervision approach with labels alone.

MEMEX Extractors

We successfully created a variety of accurate, rapidly-executing extractors in DeepDive to support the goals of the MEMEX program. Since DeepDive was commercialized as Lattice and acquired by Apple in 2017, the team has reconstituted the MEMEX extractors using Snorkel, and in August 2018 gave a successful demonstration of both the extractors and front-end user interface to personnel at USDOJ. As future work, we intend to pursue deploying these extractors to support law enforcement via a front-end interface designed

to enable rapid interaction with large KBs populated using the weak supervision methods developed as part of this program.

5.0 CONCLUSIONS

The overall goal of the MEMEX project was to develop software systems that would enable analysts to quickly build high-quality, scalable domain-specific indexing and search engines. Under MEMEX, our team made substantial progress towards enabling analysts to leverage the power of scalable machine learning systems to accomplish a variety of domain-specific tasks with little required expertise on the underlying algorithms. Our integration of the DeepDive system with a DSL generated by Delite yielded substantial improvements in analysis runtime due to improvements in Gibbs sampling runtime, our deployment of this system enabled the creation of effective extraction pipelines for the MEMEX project, and our resultant effort towards leveraging weak supervision to support analytics workloads has yielded systems such as Snorkel that are deployed in support of critical functions in government, industry, and academia. While DeepDive was commercialized as Lattice and acquired by Apple in 2017, Snorkel has become a widely-used open source software available on GitHub. The MEMEX project has supported the generation of over 30 peer-reviewed technical publications.

6.0 PUBLICATIONS SUPPORTED BY MEMEX

1. Ratner, A., Hancock, B., Dunnmon, J., Goldman, R., and Ré, C., "Snorkel MeTaL: Weak Supervision for Multi-Task Learning," *Proceedings of the 2nd Workshop on Data Management and End-to-End Machine Learning*, June 2018.

2. Hancock, B., Varma, P., Wang, S., Bringmann, M., Liang, P., and Ré, C., "Training Classifiers with Natural Language Explanations," *Proceedings of the 56th Annual Meeting of the Association of Computational Linguistics*, June 2018.

3. Aberger, C.R., Lamb, A., Olukotun, K. and Ré, C., "LevelHeaded: A Unified Engine for Business Intelligence and Linear Algebra Querying," *ICDE, 2018.*

4. Ratner, A., Bach, S., Ehrenberg, H., and Ré, C., "Snorkel: Fast Training Set Generation for Information Extraction," *Proceedings of the VLDB Endowment*, 2018.

5. Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., and Ré, C.,"Fonduer: Knowledge Base Construction from Richly Formatted Data," *Proceedings of the VLDB Endowment*, Vol. 10, No. 11, pp. 1190–1201, 2018.

6. Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C. and Zaharia, M., "DAWNBench: An End-to-End Deep Learning Benchmark and Competition." *Training*, Vol. 100, No. 101, pp.102, 2017.

7. Prabhakar, R., Zhang, Y., Koeplinger, D., Feldman, M., Zhao, T., Hadjis, S., Pedram, A., Kozyrakis, C. and Olukotun, K., "Plasticine: A Reconfigurable Architecture for Parallel Patterns." *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 389-402, June 2017.

8. Varma, P., He, B.D., Bajaj, P., Khandwala, N., Banerjee, I., Rubin, D. and Ré, C., "Inferring Generative Model Structure with Static Analysis." *Advances in Neural Information Processing Systems*, pp. 240-250, 2017.

9. Ratner, A., Ehrenberg, J., Hussain, Z., Dunnmon, J., and Ré C., "Learning to Compose Domain-Specific Transformations for Data Augmentation." *Advances in Neural Information Processing Systems*, pp. 3239–3249, 2017.

10. De Sa, C., Ratner, A., Ré, C., Shin, J., Wang, F., Wu, S., and Zhang, C., "Incremental knowledge base construction using DeepDive." *Proceedings of the VLDB Endowment*, No. 26, Vol. 1, pp. 81-105, 2017.

11. Rekatsinas, T., Chu, X., Ilyas, I.F. and Ré, C., "Holoclean: Holistic data repairs with probabilistic inference." *Proceedings of the VLDB Endowment*, No. 10, Vol. 11, pp.1190-1201, 2017.

12. Bach, S.H., He, B., Ratner, A. and Ré, C., "Learning the Structure of Generative Models without Labeled Data." *International Conference on Machine Learning*, pp. 273-282, 2017.

13. Ratner, A.J., De Sa, C.M., Wu, S., Selsam, D. and Ré, C., "Data programming: Creating large training sets, quickly." In *Advances in Neural Information Processing Systems*, pp. 3567-3575, 2016.

14. He, B.D., De Sa, C.M., Mitliagkas, I. and Ré, C., "Scan Order in Gibbs sampling: Models in Which it Matters and Bounds on How Much." In *Advances in Neural Information Processing Systems*, pp. 1-9, 2016.

15. Ehrenberg, H.R., Shin, J., Ratner, A.J., Fries, J.A. and Ré, C., "Data programming with DDLite: putting humans in a different part of the loop." *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 13, 2016.

16. Aberger, C., Lamb, A., Olukotun, K., and Ré, C., "Mind the gap: bridging multi-domain query workloads with Emptyheaded." *Proceedings of the VLDB Endowment*, Vol. 10, No. 12. pp. 1849–1852, 2017.

17. Aberger, C., Tu, S., Olukotun, K., and Ré, C., "EmptyHeaded: A Relational Engine for Graph Processing," *SIGMOD '16: Special Interest Group on Management of Data*, June 2016. (Best of Award)

18. De Sa, C., Olukotun, C., and Ré, C., "Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling," *ICML '16: Proceedings of the 33rd Intl. Conference on Machine Learning*, June 2016. (Best Paper Award)

19. Koeplinger, D., Prabhakar, R., Zhang, Y., Delimitrou, C., Kozyrakis, C., and Olukotun, K., "Automatic Generation of Efficient Accelerators for Reconfigurable Hardware," *ISCA '16: 43rd International Symposium on Computer Architecture*, June 2016.

20. Prabhakar, R., Koeplinger, D., Brown, K., Lee, H., De Sa, C., Kozyrakis, C., and Olukotun, K., "Generating Configurable Hardware from Parallel Patterns," *ASPLOS '16: 21st International Conference on Architectural Support for Programming Languages and Operating Systems*, April 2016.

21. Brown, K., Lee, H., Rompf, T., Sujeeth, A., De Sa, C., Aberger, C., and Olukotun, K. "Have Abstraction and Eat Performance, Too: Optimized Heterogeneous Computing with Parallel Patterns," *CGO '16: International Symposium on Code Generation and Optimization*, March 2016.

22. De Sa, C., Zhang, C., Ré, C., and Olukotun, K., "Taming the Wild: A Unified Analysis of Hogwild!-Style Algorithms," *NIPS '15: Proceedings of the 28th Neural Information Processing Systems Conference,* December 2015.

23. De Sa, C., Zhang, C., Ré, C., and Olukotun, K., "Rapidly Mixing Gibbs Sampling for a Class of Factor Graphs Using Hierarchy Width," *NIPS '15: Proceedings of the 28th Neural Information Processing Systems Conference,* December 2015.

24. De Sa, C., Olukotun, K., and Ré, C., "Global Convergence of Stochastic Gradient Descent for Some Non-convex Matrix Problems," *ICML '15: Proceedings of the 32nd Intl. Conference on Machine Learning*, July 2015.

25. Rompf, T., Brown, K., Lee, H., Sujeeth, A., Olukotun, K., "Go Meta! A Case for Generative Programming and DSLs in Performance Critical Systems," *SNAPL 2015: Symposium on Advances in Programming Languages*, May 3–4, 2015.

26. McAfee, L. and Olukotun, K. "EMEURO: A Framework for Generating Multi-Purpose Accelerators via Deep Learning," *CGO '15: International Symposium on Code Generation and Optimization,* San Francisco, CA, Feb. 10, 2015.

27. Mallory, E., Zhang, C., Ré, C., and Altman, R., "Large-scale extraction of gene interactions from full-text literature using DeepDive," *Bioinformatics*, Vol. 32, No. 1, pp. 106-113, 2016.

28. Shin, J., Ré, C., and Cafarella, M. "Mindtagger: a demonstration of data labeling in knowledge base construction." *Proceedings of the VLDB Endowment*, Vol. 8, No. 12, pp. 1920-1923, 2015.

29. De Sa, C., Zhang, C., Olukotun, K. and Ré, C. "Taming the wild: A unified analysis of hogwild-style algorithms." In *Advances in Neural Information Processing Systems*, pp. 2674-2682, 2015.

30. Sujeeth, K., Brown, K., Lee, H., Rompf, T., Chafi, H., Odersky, M. and Olukotun, K. "Delite: A Compiler Architecture for Performance-Oriented Embedded Domain-Specific Languages," *TECS'14: ACM Transactions on Embedded Computing Systems*, July 2014.