Research Statement

Maeda F. Hanafi, Ph.D.

1 Introduction

With the rise of data analytical and artificial intelligence (AI) techniques in production and industry usage, harnessing the latest tool often requires technical expertise. My research goal is to make these techniques accessible to all end-users by targeting challenges around their usage.

My past research focused on building user-friendly tools for building data extraction programs and debugging data processing pipelines:

- **Building Data Extraction Tools** How should a tool be designed for a non-technical user to quickly and accurately extract data without any programming? (Section 2) How does a tool facilitate data extraction for document types beyond text, especially print documents or PDFs? (Section 3)
- **Debugging Data Processing Pipelines** In a much larger scenario involving many data processing modules, what should a tool visualize to a user to accurately identify the source of the errors in the outputs? (Section 4)

The tools I have built utilize *interpretable and transparent models* [5, 3]. These are models include using rules, scripts, or some domain specific language. The increasing usage of such interpretable models is inspired by the need to explain and interpret a machine's reasons for arriving at its decisions and predictions to users, especially with the increasing reliance on machines to provide data-driven decisions [2]. Providing the reasons for a machine's outputs is not always straightforward due to the complex and inherent structure of some models, such as neural networks, which coined the term, "black-box models". Some approaches have studied the idea of generating *explanations* of the model's outputs [16]. In Section 2 and 3, the tools synthesize data extraction rules. In Section 4, the tool synthesizes *explanations* in the form of predicates that captures patterns found only the error outputs of a data processing pipeline.

Additionally, the tools I have built integrated the following ideas: (1) *mixed-initiative interfaces* [10], where either the end-user or the system contributes to the data task, e.g. extracting or debugging, whenever possible, and (2) *program synthesis* [6], which provides automation by automatically generating programs from user-specifications of how the program should behave.

My future research builds on top of the interdisciplinary approach of my past research. My aim is to go beyond using interpretable, transparent, and rule-based models, and target challenges in much more complex, black-box models, which are prevalent in AI. This research statement will briefly describe my past research work, followed with my future research goals.

2 Auto-Generating Extraction Rules from User-Highlighted Texts with SEER

SEER helps non-technical end-users extract data by automatically synthesizing easy-to-understand extraction rules from user-highlighted text examples. End-users such as journalists, scientists, and subject matter experts (SMEs), who wish to encode their knowledge in the extraction rules may not necessarily know how to program in traditional languages such as Python or regular expressions. SEER only requires the user to highlight examples of the text they wish to extract. SEER then automatically generates easy-to-understand extraction programs in Visual Annotation Query Language (VAQL), a standard visual query language for information extraction [13]. To guide the search for extraction programs to suggest to the user, SEER ranks and selects programs according to how actual developers build extraction programs manually. My

collaborators and I confirmed our method of ranking rule primitives with our user study on SEER [7]. Our user studies proved SEER effective in helping end-users quickly and accurately complete data extraction tasks without any programming. SEER is integrated at IBM's text analytics tools, BigInsights [1].

3 Identifying Structure from PDFs with Texture

While SEER is limited to extracting over text documents, TEXTURE is a framework designed to handle extraction over print and PDF documents through structure identification. We observed that structure is integral in improving the accuracy of the extraction programs. For instance, suppose one has an extraction program to extract all the universities in a corpus of resumes as an entity recognizer would. Extracting all of the candidate's attended universities from the educational history of resumes may result in many incorrect extractions as traditionally text-based extractors such as the entity recognizer or even rules in SEER cannot differentiate between universities in the educational history section versus universities listed under work experience. Structures, such as sections, lists, paragraphs, titles, etc., define the context boundaries in print documents. Structures in print documents are visually segmented by white spaces, margins, or lines, while the context boundaries of text-based documents are typically defined by sentence structures.

Thus, TEXTURE follows a two-step process for text extraction over print documents:

1. Structures are identified with heuristics, which the user selects from the heuristic repository in TEXTURE or develops the heuristic herself. Heuristics range from very simple rules to complex and well-researched approaches for specific structure identification.

2. Text is then extracted through TEXTURE 's Structure-Based Extraction Language (SBEL), where SBEL rules can refer to the identified structures to pinpoint the targeted text.

Through our qualitative evaluation, students developing within TEXTURE's framework, wrote structure identification heuristics with high precision and recall [9].

4 Debugging Large Data Processing Pipelines with WhyFlow

While SEER and TEXTURE focuses on the challenges of building data tools to assist end-users for data extraction, WHYFLOW focuses on debugging the outputs of such tools, specifically *data extraction pipelines*. A data extraction pipeline processes input data sources through a data flow that consists of code modules that extract, transform, and output the data. A data flow program is an acyclic graph, where each node is an operator that takes in data and outputs data.

Errors from code modules inevitably arise and include unintended use of attributes, unexpected schema changes, incorrect calculations. Errors surface in the output as incorrect attribute values, NULL values, or unexpected output records. However, debugging data flows requires backtracking the provenance of sampled error datapoints to compare against the expected datapoints. Comparing error and expected datapoints at the output of every module is iterative, manual, and time-consuming. The error manifests differently at each module's output, and the debugger risks misidentifying the actual source of the errors.

To assist the end-user in accurately identifying the source of errors in a data flow, WHYFLOW interactively *explains* the errors in a data flow, as outlined below:

1. Labeling Final Outputs: Inspecting the final outputs of a data flow, an end-user labels the datapoints that she deems as errors. Such datapoints can have unexpected attribute values, such as NULL values or values beyond an expected, known threshold.

2. *Back-propagation*: Given the user-identified errors on the final output, WHYFLOW labels upstream datapoints that may have contributed to the errors at the final outputs using a back-propagation technique [8].

3. Explaining Errors: After obtaining the error and non-error datapoints for all modules in the entire provenance, WHYFLOW synthesizes explanations of the errors in a data flow. At a given code module's outputs, explanations are generated for its outputs, essentially capturing patterns that differentiate error datapoints from non-error datapoints.

Our user evaluations show that WhyFlow, through a combination of its error explaining predicates and effective data visualizations, enables most users to accurately identify the source of the faults [8].

5 Research Vision

My future research builds on top of the ideas from the developing SEER, Texture, and WhyFlow. Much of the focus revolved around using interdisciplinary techniques from program synthesis, human-computer interactions, and databases to work on challenges in building and debugging data processing tools. I plan on working on challenges in AI deployments, including *explainable AI* (XAI) [18]. XAI focuses on the challenge of "opening up the black-box" by explaining the AI's outputs to the end-user in order for a much more collaborative human-machine interaction framework. The rest of the document briefly describes the core challenges of my research interests.

- *Codifying principles for visualizing the machine decision logic to the end-user.* Visualizing the machine's logic behind its outputs helps increase the user's understanding and consequently its trust to the system. The machine's logic is often influenced by several factors. But how do we effectively visualize the logic of the machine to the user? Existing works include using *interpretable models*, such as decision trees or rules [5, 21], and visualizing the factors influencing the results [17]. Challenges include visualizing the machine decision logic for much more black-box approaches [11, 4]. Another interesting direction would be to also outline principles of visualizing the machine decision logic that are specific for human-machine collaboration in dynamic, time-critical environments, such as warfare scenarios, gunfire, or aircraft training simulations.

- *Effectively communicating the machine's learned boundaries to the end-user.* Existing approaches include Wachter et al. proposal of *counterfactuals* [19]. A counterfactual is a "statement of how the world would have to be different for a desirable outcome to occur", e.g. you need at most income x to qualify for governmental financial aid. A counterfactual can be compared with a prediction of interest to figure out the decision boundaries of the model. With the widespread usage of AI involving millions of datapoints, communicating the learned boundaries must go beyond a per datapoint view. Given that a model is only limited to whatever data it was trained on, challenges of designing a more effective visualization include allowing the user to easily identify the characteristics of the datapoints the AI was not trained on in the first place.

- *Codifying Principles of Synthesized Artifacts.* There are existing works on *synthesizing artifacts*: visualizations[14, 15], extraction rules[7, 9, 12], explanations[20, 8]. It is important to communicate the logic behind the synthesis of these artifacts to the user, regardless of whether they have technical expertise or not. This research direction should also aim to codify principles that deal with black-box models and how one would optimize for interpretability and explainability [5].

References

[1] Ibm biginsights. `https://www.ibm.com/support/knowledgecenter/en/SSPT3X/SSPT3X_welcome.html`. Accessed: 2021-01-21.

[2] 2018 reform of eu data protection rules, 2018. `https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf`.

[3] L. Chiticariu, Y. Li, and F. R. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! EMNLP 2013, pages 827–832, 2013.

[4] Y. Chung, T. Kraska, N. Polyzotis, and S. E. Whang. Slice finder: Automated data slicing for model validation, 2018.

[5] A. A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, Mar. 2014.

[6] S. Gulwani. Dimensions in program synthesis. In *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming*, pages 13–24. ACM, 2010.

[7] M. F. Hanafi, A. Abouzied, L. Chiticariu, and Y. Li. Seer: Auto-generating information extraction rules from user-specified examples. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6672–6682, New York, NY, USA, 2017. ACM.

[8] M. F. Hanafi, A. Abouzied, M. Danilevsky, and Y. Li. Whyflow: Explaining errors in data flows interactively. In *In Workshop on Data Science with Human in the Loop (DaSH 2020)*, 2020.

[9] M. F. Hanafi, M. Mannino, and A. Abouzied. A collaborative framework for structure identification over print documents. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA'19, New York, NY, USA, 2019. Association for Computing Machinery.

[10] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page 159–166, New York, NY, USA, 1999. Association for Computing Machinery.

[11] S. Krishnan and E. Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*, HILDA'17, pages 4:1–4:6, New York, NY, USA, 2017. ACM.

[12] V. Le and S. Gulwani. Flashextract: A framework for data extraction by examples. PLDI '14, pages 542–553, 2014.

[13] Y. Li, E. Kim, M. A. Touchette, R. Venkatachalam, and H. Wang. Vinery: A visual ide for information extraction. *Proc. VLDB Endow.*, 8(12):1948–1951, Aug. 2015.

[14] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, Apr. 1986.

[15] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448, 2018.

[16] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Vaughan, and H. Wallach. Manipulating and measuring model interpretability, 2018.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[18] G. Vilone and L. Longo. Explainable artificial intelligence: a systematic review, 2020.

[19] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2017.

[20] T. Wu, M. T. Ribeiro, J. Heer, and D. Weld. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy, July 2019. Association for Computational Linguistics.

[21] Y. Yang, E. Kandogan, Y. Li, P. Sen, and W. S. Lasecki. A study on interaction in human-in-the-loop machine learning for text analytics. In *IUI Workshops*, 2019.