1 Introduction of research area

My work entails static program analysis, and more specifically dataflow analysis. My focus so far has been on modularization in the design of dataflow analyses, and on incrementalization of fixpoint-calculations so as to improve scalability of analyses in written using reference attribute grammars.

For modularization, I wrote a framework that alllows independent specification of dataflow analyses and their design parameters in datalog.

2 Two concepts from lectures

2.1 Tension between formal methods and large systems

The tension between formal methods and large systems is front and center in my area: Much of the current research is concerned particularly with scalability. Many of the methods are intractable for larger code bases, and those that are tractable need to give up on precision. Much of the research I have seen aims at reducing the amount of precision one needs to give up for the scalable algorithms.

2.2 Design by contract

Violations of contracts may be detected by static analyses deducing the properties used in contract specification. For example, a null pointer analysis might emit a warning if an assertion requires some possibly-null pointer to be non-null.

It is also relevant to my work in that the analyses I write may require certain structure from its input data. For example, standard data flow analyses require that the deduced property forms a semilattice.

3 Two concepts from guest lectures

3.1 Problem space vs solution space

The conceptualization of problem space and solution space, outside of being visually pleasing, maps closely to how we think about research questions versus the actual work. Keeping this distinction in mind is helpful in avoiding time drains by not veering offp| into side quests outside of the solution space. It may also be helpful as a source of creativity: When you have noticed that you have moved outside of the solution space, you may ask yourself if you

have ended up in an interesting place in the problem space instead. This may give rise to new research questions.

3.2 Stakeholders

Thinking about stakeholders may also improve the quality of the performed work. It encourages thinking about if the work I'm doing actually matters, or if I am effectively throwing away time and money on masturbatory projects. In my specific work, this would entail whether what I do would be likely to assist in the robustness and reliability of real software systems.

4 Data Scientists versus Software Engineers

4.1 Are the essential differences presented true?

I have not worked much with data scientists, so whether I agree or not will be mostly based on speculation and my imagination of data scientists do.

Data science looks like another type of software engineering: It is an area of study on how to construct computing systems that can perform some given task. To the extent that there is a lack of concrete systems thinking among practitioners in the area, this seems more to be because of the general myopic tendency associated with specialization. Such tunnel vision improves with experience, so if there has been an influx of data scientist juniors it would explain the apparent a lack of experience with engineering principles.

4.2 Will the roles keep differentiating or merge?

I believe that they will merge in the sense that data scientists will start consciously associating their work as part of software engineering. As effective software development requires some understanding of the systems one interacts with, the real world practice of data science will learn from the rest of the system and vice versa. As such, it will remain differentiated from software engineering in general while also finding its harmonious place inside of it.

5 Paper analysis

5.1 MLScent: A tool for Anti-pattern detection in ML projects

5.1.1 Core ideas and importance to software engineering

ML systems pose code smells/anti-patterns/maintenance defects with novel characteristics.

The paper presents a static analyzer aiming at detecting these issues.

The static analyzer is implemented as a set of AST node filters, called *detectors*. Whenever a detector matches a node, it emits a warning. Some of the detectors are specific to certain python libraries commonly used in machine learning.

The importance of this paper with respect to software engineering lie in the general idea of providing better static analyses by leveraging domain-specific knowledge.

5.1.2 Relation to my research

This relates to my research to the extent that it implements static analyses. However, from the presentation the actual analyses seem to be quite simple.

5.1.3 Integration into a larger AI intensive project

These techniques would be usefully integrated in continuous integration as part of the code review process. This would keep lazy developers from getting sloppy in their practices.

My own research could fit into this project by providing the framework for writing the analyses.

5.1.4 Adaption of my research

I might adapt the type of analyses that I work with to take into account more domain-dependent information. With respect to the area of machine learning, some ideas would be:

- Dataflow analysis to track potential leakage of internal data through models with access to multiple data sources.
- Tracking system impact of updating models. For example, when a certain model is updated, it might be of interest to know what other models in the system make use of data that this model produces. This

type of analysis would take some steps to address the issue of entanglement, an issue explained in Hidden Technical Debt in Machine Learning Systems by D. Sculley et al..

- Tracking which models have been indirectly fitted to other models in the environment during training. This analysis could be executed during training to store such dependency information in the model, providing another source of dependency in addition to the one previously mentioned.

In greater generality, I could consider what areas of computing perform complex tasks while lacking basic sanity checking tools. This need not be machine learning, but could also be bioinformatics, industrial chemical processing, social network modelling, world systems simulations, and so on.

5.2 A Taxonomy of Architecture Options for Foundation Model-based Agents: Analysis and Decision Model

5.2.1 Core ideas and importance to software engineering

A taxonomy of concepts for classifying ML systems are put forth. This taxonomy contains categories such as input modalities, what external interfaces it has access to and level of autonomy. The concepts provide a standardised language for classifying architectures for machine learned agents.

The taxonomy is further used to create a decision model for the design of an agent.

The general approach of creating a classification scheme for components of a particular domain of software development, and using that taxonomy to provide guidelines for developing those components, is generalizable to other domains.

This particular conceptual framework might be useful for thinking about what type of static analyses might be designed geared towards applications of data science, something I discussed in the analysis of the previous paper. One could also consider whether it is possible to automatically perform this classification on already existing agents through cleverly designed analyses.

5.2.2 Relation to my research

This is not really related to my research at all, I just found it interesting because it could possibly illuminate some of what all the machine learning fuzz is about.

5.2.3 Integration into a larger AI intensive project

Scenario: Create an agent fetching relevant documentation from manuals for information that is not easy to look up, such as information about syntactical constructions.

This paper could be used in a larger project by providing an initial architectural plan for the agent.

My research would not find into the project, as it is not related.

5.2.4 Adaption of my research

If I were to implement the adaptions I mentioned in the previous paper analysis, this paper could provide the foundations for formulating new analyses aimed at machine learning agents.

To further the aims of this paper, a static analysis could deduce some of the classifications. For example, it could find the possible input modalities by recognizing what data could be fed the agent. It could also find out what the actual capabilities of it are.

In the other direction, the paper could further the area of program analysis by providing the basic terminology for domain specific analyses aimed at data science. This could potentially be used to form checks specialized to different types of agents as well. Such specialization could include

- Ensuring correct formats of input data depending on modality.
- Ensuring that the agent may only impact reality through ports fitting to its mode of interacting with the world.

If one also integrates agents into the process of deciding where in the static analysis to sink more computational resources, the decision procedure might come in handy when actually implementing the agent. Furthermore, even in the the initial ideation phase, the terminology might be useful to even know which dimensions the design space of an agent might have.

6 Research Ethics and Synthesis Reflection

6.1 Search and screening process

I looked through the lists of articles, checking the ones that seemed kind of interesting. From those, I picked the ones whose abstract looked most interesting.

6.2 Pitfalls and mitigations

I did not really find an misleading abstracts. At most, I found that some papers leaned into the aspects of the abstract I found more boring than interesting.

6.3 Ethical considerations

I did a quick background checks on the authors and their papers, and there didn't seem to be any great complaints.