Crafting Research Statements for Faculty Job Applications

Anil Rustgi, MD
Professor of Medicine
Director, Herbert Irving Comprehensive Cancer Center

Sandra Ryeom, PhD
Associate Professor of Surgery
Associate Dean for Postdoctoral Affairs at VP&S

What is the purpose of a research statement?

- Faculty search committees are looking for:
  - The trajectory of your research (grad student/postdoc)
  - Highlights of your research accomplishments (postdoc)
  - Scientific fit in their Department/Center/Institute
  - Writing ability

General Outline For Research Statements

5-10%

- Short intro to your path into biomedical research
  - Grad student work, reasons you were interested in research

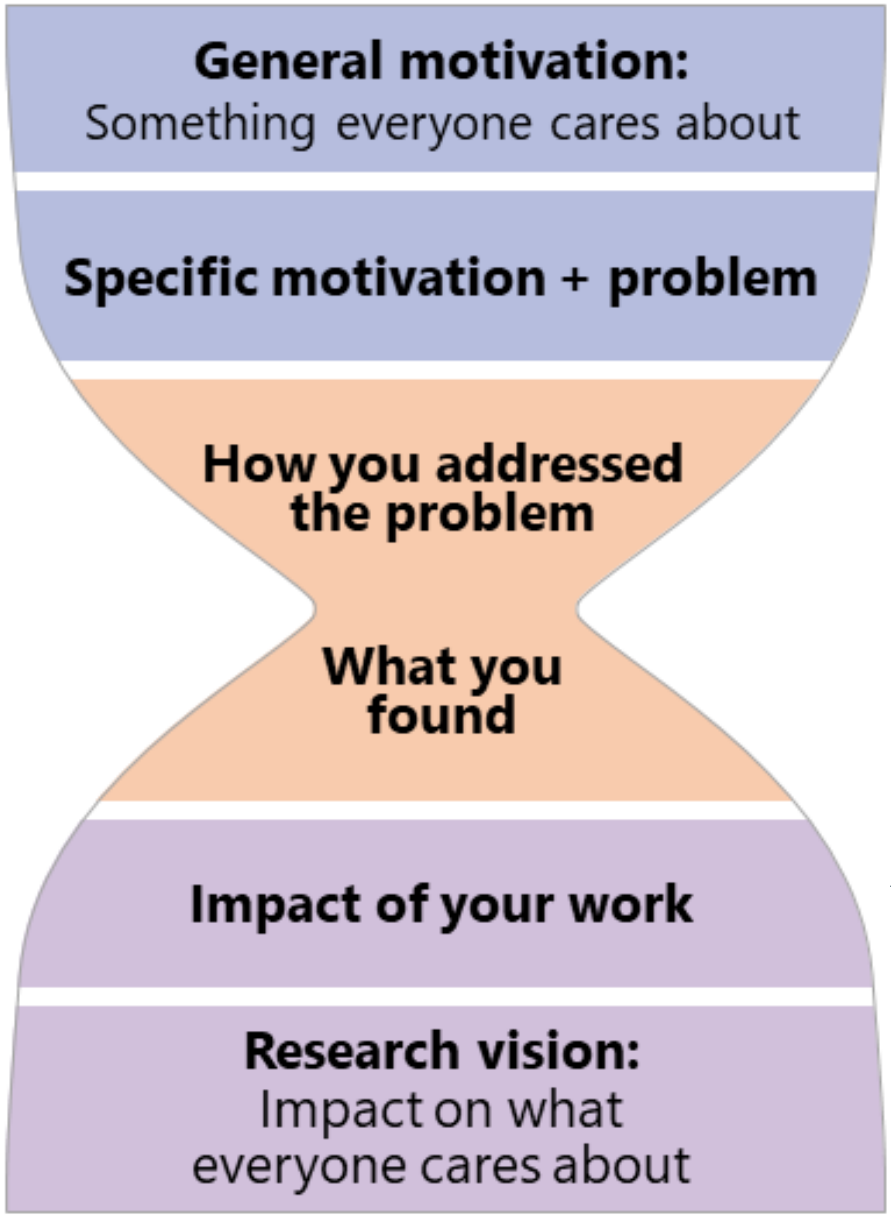
60-65%

- Describe your current research/accomplishments as a postdoc
  - How your research question addresses a gap in the field
  - Include data, cite publications
  - Discuss methodology (highlight novel methods
  - Use hourglass concept

30%

- Describe the direction you plan to pursue
  - What you will do as an independent faculty
  - How your research program fits into the department you are applying to (tailor for each department)

Research Statement Structure



Use figures here to support important results/claims

MIT website

Things to think about:

- Research statements should usually be ~2 pages (maybe 3 with figures/diagrams)
  - Make sure its readable with appropriate spacing, font size and margins
  - Divide into clearly labeled sections

- Think about the audience
  - Is it a department with a wide range of expertise or is it a focused department? (ie. Dept of Cancer Biology vs Dept of Biology)
  - Remember to cite faculty in the Department if they have made contributions to your field in the reference section

- Do they ask for a separate diversity and teaching statement?

Some Tips for Research Statements

- Do not make overly grandiose claims
  - “My work has completely revolutionized the field of xxx..”
- Refer to yourself if you did the work, cite collaborators if the work was done collaboratively but highlight your contribution
- Don’t discuss negative data
  - “I spent 2 years developing xx but unfortunately, the screen was negative..”
- Unusual situations/circumstances should be addressed in the cover letter
  - Ie. Long absence from research in your CV

Cover letter vs Research Statement

- Cover letter should by 1 page and include:
    - the position you are applying for;
    - your current place of work or study;
    - your enthusiasm for the position, the department, and the institution;
    - suggests which general areas of accomplishment make you a good fit for the position
        - Ie. My research is in genome integrity complementing the research of many faculty in your department.
    - mention which documents are attached as part of the application
        - ie who your references letters will be written by

Research Proposals vs Research Statements

- Some faculty positions will ask for a research proposal which is usually a specific aims page of a future R01 grant application
  - Can include significance/impact of your R01

- Research proposals are **different** from research statements

Weak Research Statements

- Overly ambitious proposals/goals as an Assistant Professor
- Lack of clear direction
- Lack of big-picture focus
- Inadequate attention to the fit within the department or position
- Too much jargon that is not explained
- Your research uses the same methodology with a different tissue/cell type (ie, “one-trick pony”)
- Typos/grammatical errors

Intro To Research Statements

- Can include an abstract
- Give the committee a sense of you as an individual – say who you are (ie. I am a machine learning expert, cancer biologist, computational scientists, etc…)
- Motivation for research, early success (graduate school research)
- Path to your postdoc

Research Accomplishments

- Describe your strongest results in the most detail
- If you want to discuss multiple manuscripts, organize them into themes
- Discuss your contributions to the field
- Discuss any awards, funding received for your work
- Cite manuscripts
- Include limited/key data

Future Research Plans

- The best future plans usually build on your prior experience but are not a direct extension of your postdoctoral work
  - What new direction will your research go in?

- Preliminary data can be included if its compelling
  - Ie. A screen performed that identified a new class of genes that you will investigate

- Highlight your independence from your current PI
  - Discuss what novel area you brought to the lab or discovered during your postdoc or how your PI will let you take the program with you

Have multiple people read your research statement!

- Peers
- Mentors
- Trainees (ie. Grad students/undergrads)
- People outside your field

Annotated Research Statement (taken from MIT website)

Interactive systems for code and data demography

Elena L. Glassman

December 21, 2017

Tells who the candidate is

I am a researcher in **human-computer interaction (HCI)**, I design, build and evaluate systems for code demography, i.e., comprehending and interacting with population-level structure and trends in **large code corpora**. These systems augment human intelligence by giving users a "useful degree of comprehension in a situation that previously was too complex."[1]

Discusses impact early on to catch readers' attention

Shares a glimpse of the candidate's vision while introducing self

In my doctoral and postdoctoral work at MIT and UC Berkeley, I have used *program analysis and synthesis techniques, interactive inference algorithms, visualization principles, and theories from cognitive science* to build systems that allow people to complete existing large-scale code-related tasks more quickly and answer new questions that were previously prohibitively time-consuming to investigate (Figs. 1 and 2). For example, OVERCODE (Fig. 3) is now deployed at UC Berkeley, where teachers give code composition feedback to more than 1500 students in a few hours.[2] EXAMPLORE (Fig. 4) allows programmers, API designers, and researchers to answer questions about how API methods are actually used in the wild.[3] Now, as a fellow at the Berkeley Institute of Data Science, I am exploring how to generalize these methods beyond code to help data scientists, social scientists, journalists, and other end-users more easily work with large amounts of data and communicate their intent to machines using concrete examples.

The conceptual key to my approach is defining *task-relevant abstractions* through data-driven (1) user-centered design or (2) inference algorithms. For example, I designed EXAMPLORE's abstract API skeleton to register and align hundreds of usage examples against each other so that users can get a high-level view of a corpus without sacrificing the ability to read concrete code. This design is supported by theories of human learning, such as analogical learning and Variation Theory: showing multiple aligned examples simultaneously helps induce accurate abstractions in the user's mind. In FIXPROPAGATOR, the abstractions are inferred from data: as a programming teacher begins to fix and give feedback on buggy student code submissions, the back end infers more general, abstract code transformations to propagate fixes and relevant teacher feedback to other buggy student code.[4]

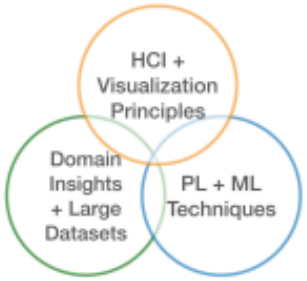


Figure 1: Common system components.

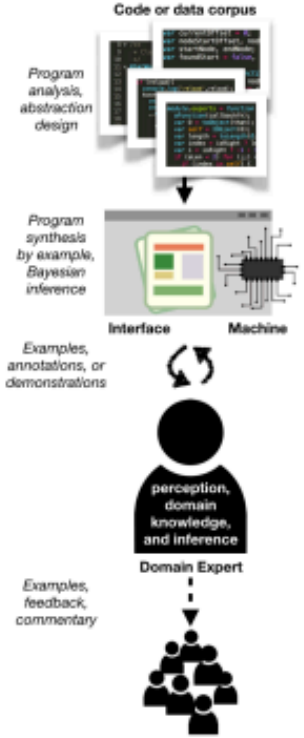


Figure 2: Common system architecture.

[1] D. C. Engelbart. Augmenting human intellect: A conceptual framework. *Stanford Research Institute. Retrieved March*, 1: 2007, 1962

[2] E. L. Glassman, J. Scott, R. Singh, P. J. Guo, and R. C. Miller. Overcode: Visualizing variation in student solutions to programming problems at scale. *ACM*

https://mitcommlab.mit.edu/eecs/wp-content/uploads/sites/6/2021/09/Elena-Glassman-research-statement-annotated.pdf

1. *Tools for teachers and students in massive classrooms*

Easily digestible motivation

Enrollment in introductory programming and data science courses is skyrocketing. Even hardware design classes can contain hundreds of students, each constructing their own simulated circuits. It is both a challenge and an opportunity to reinvent how we teach students and how students teach each other.

Quantify and show, don't tell

With collaborators and mentees, I built, user-tested, and deployed five systems at MIT and UC Berkeley that explore this space of possibilities.

OVERCODE is an example of code demography for visualizing and exploring thousands of solutions to the same programming problem. It uses both static and dynamic analysis to cluster similar solutions and represents each cluster as a synthesized solution that implicitly describes both the cluster center and its boundaries. Teachers can filter and further cluster solutions based

Title describes a switch of contexts

2. *Supporting programmers in the wild and at companies with large codebases*

Understanding the space of possible code solutions is helpful beyond the classroom, as well. EXAMPLORE (Fig. 4) is an example of code demography for visualizing and exploring thousands of code examples using the same API: in collaboration with software engineering researchers at UCLA, we created an interactive visualization by mining hundreds of thousands of open-source Github repositories to reveal the common and uncommon ways in which the open-source developer community uses a Java API method.[9] In a within-subjects lab study, we found that EXAMPLORE helped programmers understand the distribution of usage patterns of a particular API method in the wild, and answer common API usage questions accurately. I plan to make this tool available as an online resource that complements official documentation and Q&A sites. I also hope to work with engineers and researchers at companies with large proprietary codebases and APIs, like Google and Facebook, to customize this tool for their internal on-boarding and code review needs. Through initial contacts, it is clear that these companies are eager to adopt and exploit new methods and interfaces, like EXAMPLORE, to increase programmer productivity.

[7] E. L. Glassman, J. Kim, A. Monroy-Hernández, and M. R. Morris. Mudslide: A spatially anchored census of student confusion for online lecture videos. In *Proceedings of the Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1555–1564. ACM, 2015b. URL http://doi.acm.org/10.1145/2702123.2702304

[8] S. Terman. Grovercode: code canonicalization and clustering applied to grading. Master's thesis, Massachusetts Institute of Technology, 2016

[9] E. L. Glassman*, T. Zhang*, M. Hearst, B. Hartmann, and M. Kim. Visualizing api usage examples at scale. In *Proceedings of the Annual ACM Conference on Human Factors in Computing Systems*, CHI '18. ACM, 2018

Hints at future research directions right after enough context is given

3. *Collaborating with PL and ML researchers to improve the interface between people and intelligent back ends*

A title that tells a story

Theories of human concept learning, e.g., Variation Theory,[10] assert that showing people strategically diverse sets of examples will help them construct mental abstractions that generalize well. The same is true when teaching machines. In the following projects, I show how examples are mined for data-driven teaching of both humans and machines. To achieve these results, I collaborated with academic and industry researchers in programming languages (PL) and machine learning (ML) to produce systems with cutting-edge intelligent back ends and front ends that enable real-world impact.

Illustrates breadth

In my first exploration of example-based inference frameworks, specifically the interactive Bayesian Case Model (iBCM), I collaborated with iBCM's creator to build an interface on top of the model to test how well domain experts could collaborate with the machine—by choosing examples and critical features—to define statistically valid clusters that were also relevant to the ex-

[10] M. Ling Lo. *Variation theory and the improvement of teaching and learning*. Göteborg: Acta Universitatis Gothoburgensis, 2012

Future directions:

How can we more naturally communicate our intent to machines?

A precise and clear question as a title

The critical, possibly latent, structural features that distinguish examples from each other help both humans and machines learn concepts and infer abstractions that generalize well. Examples are, therefore, a natural medium for communication between people and machines. Code and data demography strongly support this kind of communication: it exposes the population-level characteristics of large datasets in a way that keeps concrete examples front and center instead of hiding them behind nodes in large graphs or other abstractions. Some of the systems I hope to build next will combine example-based inference techniques with data demography to help users thoroughly examine the data in their corpus (1) before selecting examples to teach the machine and (2) while reviewing the machine-inferred results. I expect that data demography will help users clarify their own intentions[15] and curate better examples to teach machines, much like OVERCODE and FOOBAZ helped teachers understand the state of their massive classroom and pick better examples to address student misconceptions. I hope to continue working with experts in programming languages and machine learning to improve the ways in which humans and machines communicate and collaborate, in order to develop applications with real-world impact.

[15] V. Le, D. Perelman, O. Polozov, M. Raza, A. Udupa, and S. Gulwani. Interactive program synthesis. *CoRR*, abs/1703.03539, 2017. URL http://arxiv.org/abs/1703.03539

A descriptive title, straight to the point

When the system fails, how do we debug?

As powerful as communicating with machines by example, annotation, and demonstration can be, existing systems can fail in opaque and frustrating ways. How do we build systems that *explain their failure* to synthesize a program that is consistent with user-provided examples? A simple failure mode in program synthesis occurs when the synthesizer's grammar does not match the user's mental model: perhaps the user is teaching the machine how to extract something from raw HTML but the grammar has no notion of matching start and end tags. The program synthesis toolkit my systems currently use, the Microsoft PROSE SDK, requires an expert-designed grammar to perform well on new tasks and corpora. Rather than co-create abstractions in a fixed grammar, what if the machine and the end-user could co-create, or at least modify, the grammar itself? As a first step, I hope to create visualizations and interaction mechanisms that demystify the internal state of program synthesis engines when they fail. I will continue collaborating with programming language researchers in academia and industry, such as my colleagues on the PROSE SDK core developer team at Microsoft Research, to create better inspection, debugging, and interaction tools for end-users and developers alike.

Tying together past experience and future work

[17] E. L. Glassman, A. L. Des
benkin, M. Cutkosky, and
Region of attraction estir
perching aircraft: A lyapu
exploiting barrier certificat
and Automation (ICRA), 201
national Conference on, pag
IEEE, 2012