Reading a (CS or EE) Research Paper

Philip Levis
CS114/CS214
Spring 2022

In this class, you'll read research papers on a wide variety of subfields of computer science and electrical engineering. You'll see that these papers vary tremendously in their terminology and assumptions, and even their structure. Some of you are familiar with many of these formats and structures: you've read hundreds of research papers from a variety of fields. Others might be reading research papers for the first time. This document is intended to give you an introduction to how to approach papers generally. Read this before you read the first paper, and refer to it continually thought the quarter: you'll find it helps a great deal. I've emphasized a few sentences as important signposts. They are:

1. *Each of these papers can teach you a lot; it is worth your time to understand them well.*
2. *If you have a question about a particular sentence or point in the paper, spend some time (5-10 minutes) trying to figure it out yourself.*
3. *Print out the paper and take handwritten notes as you read; don't take notes by typing.*
4. *Read the paper in multiple passes.*
5. *When reading papers from a research field you're not familiar with, you should carefully read the abstract and spend a good deal of time (e.g., an hour) understanding the introduction.*
6. *In the context of this class, you should focus on prior work more than related work.*
7. *While you read technical and evaluation sections, think not just about what they say, but also about what they don't say.*

Spend 5 Hours Reading

I expect you to spend 5 hours reading and understanding each paper and 1 hour writing your summary. If the paper is from your field and you can read it much faster, great! The principal learning in this course is lecture, reading, and writing.

Each of these papers can teach you a lot; it is worth your time to understand them well. You will begin to see reflections of the ideas in them in many technical situations.

For example, in 2021 I taught a class in which students read Ph.D. dissertations of computer architects from the late 80s and early 90s (the "golden age" of architecture). One dissertation was about using asynchronous timing so processors wouldn't have to be conservative in their timing based on temperature, instruction complexity, or process variations: rather than a fixed clock speed, the clock would tick based on how long it actually took to execute the instructions. The idea never took off, but two weeks later a student was interviewing at Apple and asynchronous clocks came up in a completely different area — the student nailed the interview because she'd thought about the tradeoffs and deep conceptual issues in play.

Terminology

Each paper expands our understanding of a deep field that very smart people have studied for decades. Each one therefore builds on a lot of terminology and expected knowledge. To take one example,

Oblique: Accelerating Page Loads Using Symbolic Execution assumes you know what symbolic execution is and have at least some understanding of how it works, such that the paper's claims make sense. Similarly, Optimality of the Johnson-Lindenstrauss Lemma assumes you know what a Euclidean metric is.

Papers use this technical terminology as a form of compression. Every paper assumes an audience and the knowledge that audience will have. The paper Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center could explain what Hadoop is, how it works, and its performance tradeoffs, but that would take up a lot of space: there are scores of papers that explore and explain this topic. The paper assumes the reader is from the computer systems research community (SIGOPS) so understands these things somewhat.

In cases when more detailed or nuanced understanding is needed, a paper explains it. This more detailed explanation typically occurs in the introduction or related work, as the paper wants to explain how its contributions differ from the most closely related prior work. The Mesos paper, for example, doesn't explain the details of Hadoop but does explain how Hadoop jobs differ from MPI ones, and why schedulers for systems running MPI jobs aren't a good solution. That is, it explains why you need something like Mesos.

In this course, each paper you read has an introductory lecture, in which a faculty member who's knowledgable about the field will explain much of the intellectual background and terminology. Paying close attention in these lectures will be extremely valuable: the information you learn will more than save you that amount of time reading the paper. You can spend an hour trying to learn the performance tradeoffs and constraints of Hadoop on your own or pay attention for 10 minutes of lecture.

If you have a question about a particular sentence or point in the paper, spend some time (5-10 minutes) trying to figure it out yourself. This doesn't mean searching the web — it means thinking and re-reading parts of the paper and checking if the related references clearly explain it. If you can't figure it out, please bring it up in the Ed discussion group. I promise you are not alone and other people are wondering the same thing. Asking good questions can earn you a small amount of credit in the class; answering questions can earn you more. But immediately defaulting to asking someone else will never exercise your mental muscles in figuring it out yourself.

How to Read

While you are reading, take notes. Print out the paper and write on it. Underline sentences. Write in the margins. Use a notebook or pad on the side to take further notes.

Print out the paper and take handwritten notes as you read; don't take notes by typing. Handwritten notes lead you to reformulate and encode the knowledge, which leads you to remember it better as well as conceptually understand it. This is especially true when reading a paper, because you can stop reading to think conceptually about the material. The principal benefit of typed notes is they are much easier to edit and restructure. Typing is therefore better when you use your notes as a tool to come back and understand the material much later or want to share it with someone else. But as a mechanism to learn in the moment, handwriting is superior.

Read the paper in multiple passes. S. Keshav has an excellent description of this process, although it is oriented more towards a graduate student within a field ("Glance over the references, mentally ticking off the ones you've already read").

- Your first pass should just take a few minutes. It lets you learn the overall structure of the paper and where it's going.
- Your second pass is when you read all of the intellectual content of the paper, but do not yet check the factual claims. If there are proofs, you don't check the proofs. If there are experiments, you take the claims about what they show at face value and don't question them.
- Your third pass is critical reading. It's when you try to understand the paper from top-to-bottom. For a theoretical/mathematical paper, you check the proofs. For an experimental paper, you look at the results and critically question whether those are the *right* experiments and whether the evidence clearly shows what the paper claims.

CS114 students, you do not have to do the third reading. CS214 students, you do, and it should be reflected in your write-up.

Abstract and Introduction

When reading papers from a research field you're not familiar with, you should carefully read the abstract and spend a good deal of time (e.g., an hour) understanding the introduction. The introduction summarizes the paper. If you understand the introduction, you'll understand the context of the work, the intellectual contributions, and the value of those contributions. Returning to terminology above, if you don't understand the terminology in the abstract or introduction, you will struggle to understand the rest of the paper. In contrast, if you don't understand the terminology of a brief sentence deep within the paper, you can still probably understand the major points of the work.

When I read a systems paper in my area of expertise, I typically only spend 5 minutes on an abstract and introduction: it's only 1-2 pages and I understand all of the terminology and claims.[1] If it's outside my area of expertise (e.g., uses formal verification), then the introduction might take 30 minutes to an hour, as I chase down references to understand what each claim means. For example, I recently reviewed a paper that discussed using *mover types* as a way to reduce parallel programs to simpler forms to prove properties about them. I wasn't familiar with the exact definition of mover types, so I spent a half hour reading through the reference (from 1975) to the part that was important for this paper and understanding mover types. When I returned to the paper I was reviewing, I could understand the algorithm it described.

After you have read the introduction, you should understand the problem the paper solves and the basic idea of the solution. You might not understand how it solves the problem yet — that's the technical portion of the paper. You might not understand why the problem is important — that's about the research field its metrics of intellectual value.

Related Work, Prior Work

Some papers put this up front, others put it at the end. This varies not only by field but also topic. Some fields, such as graphics in CS (SIGGRAPH), essentially require a very detailed prior work section at the beginning of the paper. You need to very clearly and correctly overview the work that the paper builds

---

[1]Different researchers have different views on this. I personally don't value motivation or framing very much: I care about the contributions and results. If I think the contributions are valuable, they could have framed everything in a way I disagree with (no, it won't solve that problem) but I'll still argue for acceptance (because, it could solve *this* problem). Other researchers, though, think that framing and motivation is critical, because it articulates an understanding of a problem. This is why we have committees, so a variety of opinions and value judgements can reach consensus.

on and place it in that context. Other fields, such as computer systems (SIGOPS) are more flexible: if it's obvious to the audience that the result is novel (e.g., it's based on a new technology), you can put related/prior work at the end.

You can distinguish related work, which are other pieces of research that are closest to this paper, from prior work, which includes work that the paper directly builds on. The Mesos paper, for example, is almost entirely related work; the systems principles it builds on are considered basic enough today that it focuses on explaining solutions to related problem and how Mesos differs. The Plasticine paper, in contrast, discusses the prior work in the introduction as background, then discusses related work in a separate section at the end.

In the context of this class, you should focus on prior work more than related work. Understanding the paper itself is more important than understanding its context within a complex space of research results.

Technical Sections and Evaluation

Fields vary greatly in what constitutes technical contributions and an evaluation. For a theory paper, the two are entwined: the technical section is often a proof or derivation, and the evaluation is whether it's correct and has an interesting observation. In more systems or engineering papers, the two sections are quite distinct. The technical section describes how the system or technique works and is built, while the evaluation involves quantitative results either from simulation or an example implementation.

Fields also differ greatly in what they consider important in an evaluation. Some fields have standard benchmarks which you should use if they are applicable. Others want to see particular experiments which provide evidence that it handles certain common concerns. For example, in CS networking (SIGCOMM) as well as systems (SIGOPS), new networking stacks and approaches need experiments using TCP, a bedrock protocol of the Internet. This is because TCP is very complex: the absence of TCP results suggests that your new idea can't work well with TCP, which makes it of very limited use. A single TCP experiment, though, even with not great but not terrible results, means you were able to make the approach work TCP and good results might be possible.

While you read technical and evaluation sections, think not just about what they say, but also about what they don't say. Is there something you're wondering, which they just don't address? It could be because the answer is well understood in the community, the community doesn't think it's very important (which doesn't mean it isn't, it just isn't at the top of their list), or an issue with the paper.

Conclusion

Reading papers is hard: they are extremely dense technical documents that describe a way in which a group of people has pushed our knowledge forward just a little bit. They are especially hard to read in the beginning. As you read more of them, it becomes easier: you not only have background knowledge that lets you understand material faster, you also have a lot more familiarity in what to look for. But you have to start somewhere. All of the papers in this class are excellent, top-tier publications: spending time to read them will teach you a lot.