Arguing a Research Project

CS 197 & 197C | Stanford University | **Sean Liu** & Lauren Gillespie
cs197.stanford.edu | cs197c.stanford.edu

Slides adapted from previous iterations of the course by Michael Bernstein and Jingyi Li

Arguing a Research Project

CS 197 & 197C | Stanford University | **Sean Liu** & Lauren Gillespie
cs197.stanford.edu | cs197c.stanford.edu

Slides adapted from previous iterations of the course by Michael Bernstein and Jingyi Li

Administrivia

197

You all have projects and groups at this point. Yay!

Due next week:

Assignment 3 — Introduction.

Progress Report 1:

Get started on your projects: some lightweight setup work determined by your CA

197C

You have your research meeting and small group meetings set up!

Due next week:

Assignment 2, Part 4: Write up RW

Work towards your research milestone!

Administrivia

197: Project members dropped?

Don't worry! We will re-scope project + grading expectations so you're not affected.

In the rare case that you are the only member left, you may transfer to another project. Your CA will ensure a smooth transition w.r.t. to assignments.

Attendance policy

One excused absence for lecture

No need to email in advance; just don't submit your attendance on Canvas

197: One additional excused absence for section / team meetings

Email cs197@cs.stanford.edu or your CA in advance

197C: One additional excused absence for small group meeting

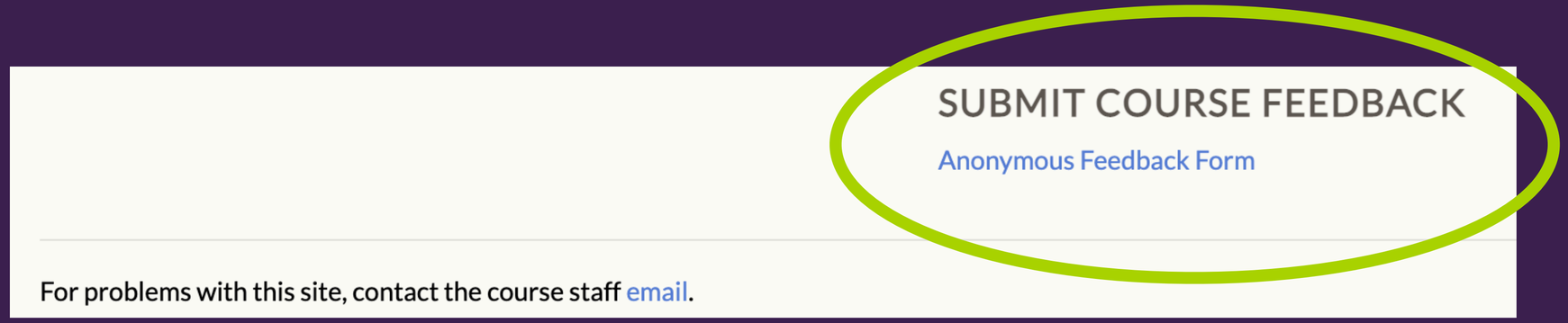
Email cs197c@cs.stanford.edu or your 197C mentor in advance

Note: We do not limit the number of absences due to **uncontrollable** circumstances (e.g., family emergencies, illnesses), but please email us ASAP

Anonymous Feedback

Option 1: High-resolution course evaluation (polled by email)

Option 2: Go to course website; scroll to bottom



FAQ: CS197

How much support will we get when implementing the projects we have selected in our group?

Intro-level projects: CAs will give you running starter code (and dataset)

Senior-level project: you will be given pointers to code; have more engineering freedom

Typically in research, you build your own prototype (advisors do not code with you). But in this class, we will give more hands-on guidance for both coding and research.

Any questions? Feel free to reach out to your CA (follow comm. guidelines)

Any questions?

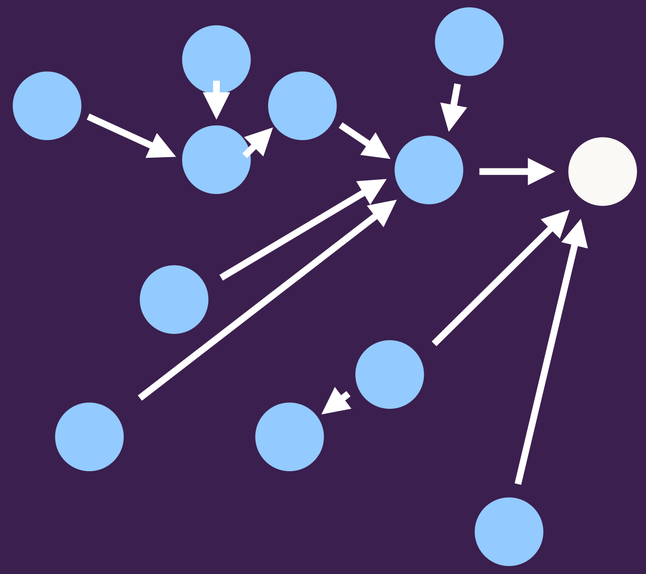
Last time

Novelty: How do we get to the point where we know what has been done, and why our idea is different, new, and exciting?

Bit flip: articulating an assumption present in all prior work that you are breaking: $X \longrightarrow X'$

Literature search process:

Iterative expansion of
the most relevant work
from the set of papers
you've seen so far



Today: from bit flip to writing a paper

How do we articulate our project persuasively to a peer? A bit flip isn't enough on its own.

If we can't explain the project clearly enough for another researcher in the same area to understand it, we don't really understand our project ourselves.

(This happens more often than you might think. It's hard!)

Your goal

Writing an Introduction to the paper. →

Often, we do this before we even start implementing the project, to make sure we can articulate it clearly.

INTRODUCTION

Crowdsourcing mobilizes a massive online workforce into collectives of unprecedented scale. The dominant approach for crowdsourcing is the microtask workflow, which enables contributions at scale by modularizing and pre-specifying all actions [7, 55]. By drawing together experts [70] or amateurs [6], microtask workflows have produced remarkable success in robotic control [48], data clustering [12], galaxy labeling [54], and other goals that can be similarly pre-specified. However, goals that are open-ended and complex, for example invention, production, and engineering [42], remain largely out of reach. Open-ended and complex goals are not easily adapted to microtask workflows because it is difficult to articulate, modularize, and pre-specify all possible actions needed to achieve them [71, 80]. If crowdsourcing remains confined to only the goals so predictable that they can be entirely pre-defined using workflows, crowdsourcing's long-term applicability, scope and value will be severely limited.

In this paper, we explore an alternative crowdsourcing approach that can achieve far more open-ended and complex goals: crowds structured like *organizations*. We take inspiration from modern organizations because they regularly orchestrate large groups in pursuit of complex and open-ended goals, whether short-term like disaster response or long-term like spaceflight [8, 9, 63]. Organizations achieve this complexity through a set of formal structures — roles, teams, and hierarchies — that encode responsibilities, interdependencies and information flow without necessarily pre-specifying all actions [15, 83].

We combine organizational structures with computational crowdsourcing techniques to create *flash organizations*: rapidly assembled and reconfigurable organizations composed of online crowd workers (Figure 1). We instantiated this approach in a crowdsourcing platform that computationally convenes large groups of expert crowd workers and directs their efforts to achieve complex goals such as product design, software development and game production.

We introduce two technical contributions that address the central challenges in structuring crowds like organizations. The first problem: organizations typically assume *asset specificity*, the ability for organization members to develop effective collaboration patterns by working together over time [83]. Clearly crowds, with workers rapidly assembled on-demand from platforms such as Upwork (www.upwork.com), do not offer asset specificity. So, our system encodes the division of labor into a de-individualized role hierarchy, inspired by movie crews [2] and disaster response teams [8], enabling workers to coordinate using their knowledge of the roles rather than their knowledge of each other.

The second problem: organizational structures need to be continuously reconfigured so that the organization can adapt as work progresses, for example by changing roles or adding teams [9, 63, 83]. Coordinating many workers' reconfigurations in parallel, however, can be challenging. So, our system enables reconfiguration through a model inspired by version control: workers replicate (branch) the current organizational structure and then propose changes (pull requests) for those

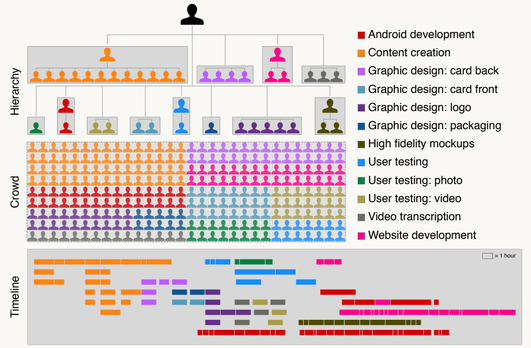


Figure 1: Flash organizations are crowds that are computationally structured like organizations. They enable automated hiring of expert crowd workers into role structures and continuous reconfiguration of those structures to direct the crowd's activities toward complex goals.

up the hierarchy chain to review, including the addition of new tasks or roles, changes to task requirements, and revisions of the organizational hierarchy itself.

Enabling new forms of organization could have dramatic impact: organizations have become so influential as the backbone of modern economies that Weber argued them to be the most important social phenomenon of the twentieth century [82]. Flash organizations advance a future where organizations are no longer anchored in traditional Industrial Revolution-era labor models, but are instead fluidly assembled and re-assembled from globally networked labor markets. These properties could eventually enable organizations to adapt at greater speed than today and prototype new ideas far more quickly.

In the rest of the paper, we survey the foundations for this work and describe flash organizations and their system infrastructure. Following this review, we present an evaluation of three flash organizations and demonstrate that our system allows crowds, for the first time, to work iteratively and adaptively to achieve complex and open-ended goals. The three organizations used our system to engage in complex collective behaviors such as spinning up new teams quickly when unplanned changes arose, training experts on-demand in areas such as medical privacy policy when the crowd marketplace could not provide the expertise, and enabling workers to suggest bottom-up changes to the work and the organization.

Framing

Bit

Network behaviors are defined in hardware, statically.

Flip

If we define the behaviors in software, networks can become more **dynamic** and more **easily debuggable**.

Project

Software-defined networking

Most important:

Focus on features that enable **dynamic** behaviors and easy **debugging**

Lower Priority:

Other technical details such as storage, speed, etc

Architecture of an Introduction

What is an Introduction?

Problem motivation

Set up the bit

Flip the bit

Instantiate the bit

Evaluation

Broader Implications

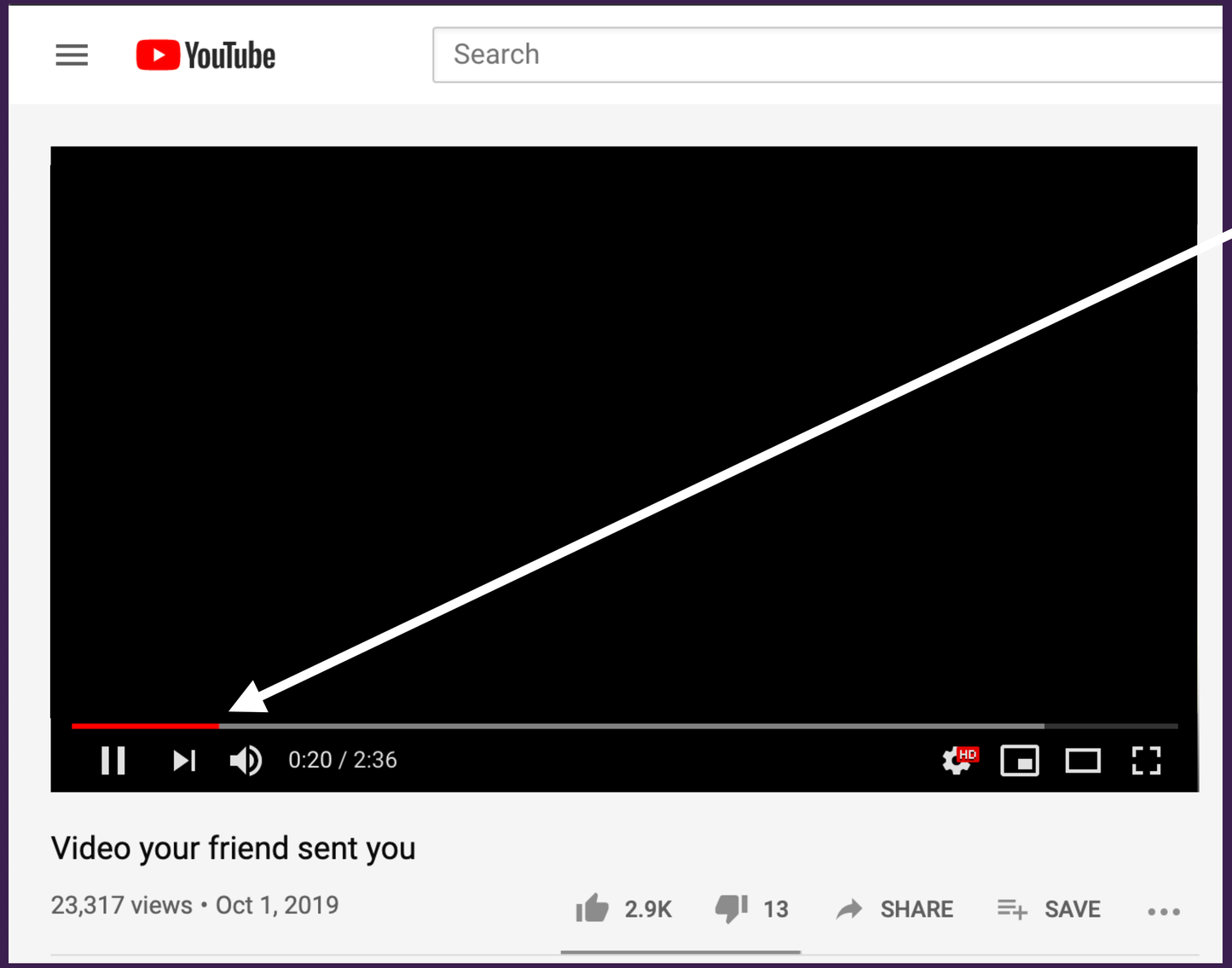
What is an Introduction?

The Introduction makes the case for your research, in brief.

Jennifer Widom:

> *"The Introduction is crucially important.* ***By the time a referee has finished the Introduction, they've probably made an initial decision about whether to accept or reject the paper*** *— they'll read the rest of the paper looking for evidence to support their decision.*
>
> *A casual reader will continue on if the Introduction captivated them, and will set the paper aside otherwise. Again, the Introduction is crucially important."*
>
> https://cs.stanford.edu/people/widom/paper-writing.html#intro

Think of it this way…



By this point, the video has hopefully made clear to you what it's about, and you've made a decision about whether to watch the rest of it.

Each introduction makes the case for two things:

1) The problem: why do we care about the problem you're solving?
2) The solution: why is your approach creative and correct?

The Problem

Turn to a partner and explain the problem that your project is working on [1 min each]

How clearly do you understand your partner's problem?

How clearly do you understand your partner's bit flip?

What is an Introduction?

Problem

Solution

Problem motivation

Set up the bit

Flip the bit

Instantiate the bit

Evaluation

Broader Implications

Each will be a paragraph.

What is an Introduction?

Problem

Problem motivation

Explain the main problem that you're trying to solve

Set up the bit

"Why isn't this problem solved yet?"

(Set up the bit you're going to flip)

What is an Introduction?

Problem

- Problem motivation
- Set up the bit

Explain the main problem that you're trying to solve

"Why isn't this problem solved yet?"

(Set up the bit you're going to flip)

Solution

- Flip the bit
- Instantiate the bit

Your high-level conceptual idea. Explain why flipping the bit is a good idea for the problem at hand.

Example

Problem

- Problem motivation — Networks are hard to (re)configure
- Set up the bit — Networks are configured in hardware and static

Solution

- Flip the bit — Software-defined networks! More dynamic and easy to debug
- Instantiate the bit

Example

Problem

- Problem motivation
- Set up the bit

Activity tracking has important health applications.

Activity tracking requires custom hardware.

Solution

- Flip the bit
- Instantiate the bit

Use just a standard cell phone for activity tracking.

More accessible and can be just as effective.

Example

Problem

- Problem motivation
- Set up the bit

Audio classification and retrieval have important applications

CLIP shows promising results on learning image, text embeddings

Solution

- Flip the bit
- Instantiate the bit

Add an audio embedding space to CLIP
Leverage the power of CLIP for audio-related downstream tasks

The Problem

Turn to a partner and explain the problem that your project is working on [1 min each]

- Problem Motivation
- Set up the bit
- Flip the bit

How clearly do you understand your partner's problem?

How clearly do you understand your partner's bit flip?

Problem

- Problem motivation — Topic sentence
- Set up the bit — Topic sentence

Solution

- Flip the bit — Topic sentence
- Instantiate the bit — Topic sentence

Tips

Problem motivation

Use citations to back up your claims about the existence of the problem, and why we should care about solving it.

Set up the bit

Use summary of related work that is in service of your bit set up

Tips

Flip the bit

The topic sentence of this paragraph is the thesis statement of your entire research project. Also known as your **research contribution**.

Pivot off of the bit you set up to flip the bit.

Explain why flipping the bit is a good idea for the problem at hand.

It should now be obvious to a reader given the prior paragraph that this research is novel, since you have proven that nobody else has flipped that bit.

Crowdsourcing platforms rely heavily on their reputation systems, such as task acceptance rates, to help requesters identify high-quality workers [22, 43]. On Mechanical Turk, as on other on-demand platforms such as Upwork and Uber, these reputation scores are derived from decentralized feedback from independent requesters. However, the resulting reputation scores are notoriously inflated and noisy, making it difficult for requesters to find high-quality workers and difficult for workers to be compensated for their quality [43, 21].

Problem = reputation is difficult to accurately derive

The rest of this paragraph gives examples to motivate the harmfulness of an unreliable reputation system.

Crowdsourcing platforms such as Amazon Mechanical Turk decentralize their workforce, designing for distributed, independent work [16, 42]. Decentralization aims to encourage accuracy through independent judgement [59]. However, by making communication and coordination more difficult, decentralization disempowers workers and forces worker collectives off-platform [41, 64, 16]. The result is disenfranchisement [22, 55] and an unfavorable workplace environment [41, 42]. Worse, while decentralization is motivated by a desire for high-quality work, it paradoxically undercuts behaviors and institutions that are critical to high-quality work. In many traditional organizations, for example, centralized worker coordination is a keystone to behaviors that improve work quality, including skill development [2], knowledge management [35], and performance ratings [58].

Set up the bit = decentralization

The rest of this introduction paragraph is dedicated to surveying related work with respect to how decentralization is architected, and to its outcomes.

To address this reputation challenge, and with an eye toward other challenges that arise from decentralization, we draw inspiration from a historical labor strategy for coordinating a decentralized workforce: *guilds*. Worker guilds arose in the early Middle Ages, when workers in a trade such as silk were distributed across a large region, as bounded sets of laborers who shared an affiliation. These guilds played many roles, including training apprentices [18, 44], setting prices [45], and providing mechanisms for collective action [52, 49]. Especially relevant to the current challenge, guilds measured and certified their own members' quality [18]. While guilds eventually lost influence due to exerting overly tight controls on trade [45] and exogenous technical innovations in production, their intellectual successors persist today as professional organizations such as in engineering, acting and medicine [46, 33]. Malone first promoted a vision of online "e-lancer" guilds twenty years ago [40], but to date no concrete instantiations exist for a modern, online crowd work economy.

flip = re-centralize via guilds

The rest of the paragraph explains the high level idea.

What is an Introduction?

Problem

- Problem motivation
  - Explain the main problem that you're trying to solve
- Set up the bit
  - "Why isn't this problem solved yet?"
  - (Set up the bit you're going to flip)

Solution

- Flip the bit
  - Your high-level conceptual idea. Explain why flipping the bit is a good idea for the problem at hand.
- Instantiate the bit

What is an Introduction?

Problem

- Problem motivation — Explain the main problem that you're trying to solve
- Set up the bit — "Why isn't this problem solved yet?" (Set up the bit you're going to flip)

Solution

- Flip the bit — Your high-level conceptual idea. Explain why flipping the bit is a good idea for the problem at hand.
- Instantiate the bit — What is the high-level architecture / system / design? How is it an instance of the bit flip?

Instantiate the bit flip

At this point, the reader understands the idea that you're proposing, but it's still very high level. In this paragraph, map that idea onto a concrete instantiation.

Typically, this is where the system or algorithm that you're creating gets a name. Explain its architecture or design at a **high level**. Make clear how this architecture or design is an instance of the bit flip.

Example

Problem

Problem motivation

Networks are hard to (re)configure

Set up the bit

Networks are configured in hardware and static

Solution

Flip the bit

Software-defined networks!
More dynamic and easy to debug

Instantiate the bit

We propose X, a network that can be dynamically updated via software, where ABC config features can be re-programmed on the fly.

Example

Problem

Problem motivation

Activity tracking has important health applications.

Set up the bit

Activity tracking requires custom hardware.

Solution

Flip the bit

Use just a standard cell phone for activity tracking.

More accessible and can be just as effective.

Instantiate the bit

We propose Ubifit, a prototype that can track walking and jogging.

Example

Problem

Problem motivation

Audio classification and retrieval have important applications

Set up the bit

CLIP shows promising results on learning image, text embeddings

Solution

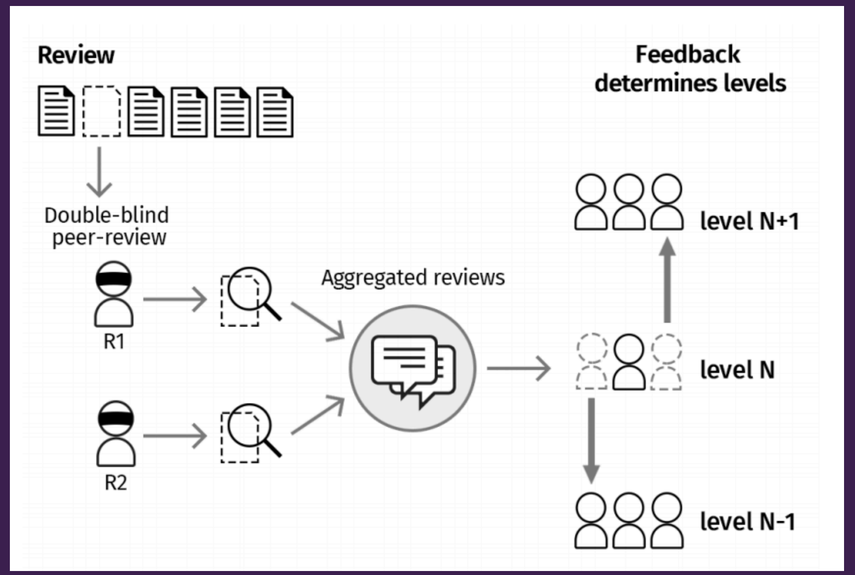
Flip the bit

Add an audio embedding space to CLIP
Leverage the power of CLIP for audio-related downstream tasks

Instantiate the bit

We duplicate the image encoder into an audio encoder and train with a new trio loss between (image, text, audio).

We present *crowd guilds*: crowd worker collectives that coordinate to certify their own members and perform internal feedback to train members (Figure 1). Our infrastructure for crowd guilds enables workers to engage in continuous double-blind peer assessment [30] of a random sample of members' task submissions on the crowdsourcing platform, rating the quality of the submission and providing critiques for further improvement. These peer assessments are used to derive guild levels (e.g., Level 1, Level 2) to serve as reputation (qualification) signals on the crowdsourcing platform. As workers gather positive assessments from more senior guild members, they rise in levels within the guild. Guilds translate these levels into higher wages by recommending pay rates for each level when tasks are posted to the platform. While crowd guilds focus here on worker reputation, our experiment implementation also explores how crowd guilds could address other challenges such as collective action (e.g., collectively rejecting tasks that pay too little), formal mentorship (e.g., repeated feedback and training), and social support (e.g., on the forums). Because



instantiation = crowd guilds system

The rest of the paragraph details how crowd guilds work.

What is an Introduction?

Problem

Problem motivation

Set up the bit

Solution

Flip the bit

Instantiate the bit

Evaluation

Broader Implications

Each will be a paragraph.

Evaluation

How did you prove that your bit flip is successful at solving the problem?

We obviously haven't covered evaluation yet in this course, so for now you'll need to take your best guess.

> How would you convince a critical reader that flipping the bit solved the problem better than the prior work?

Implications

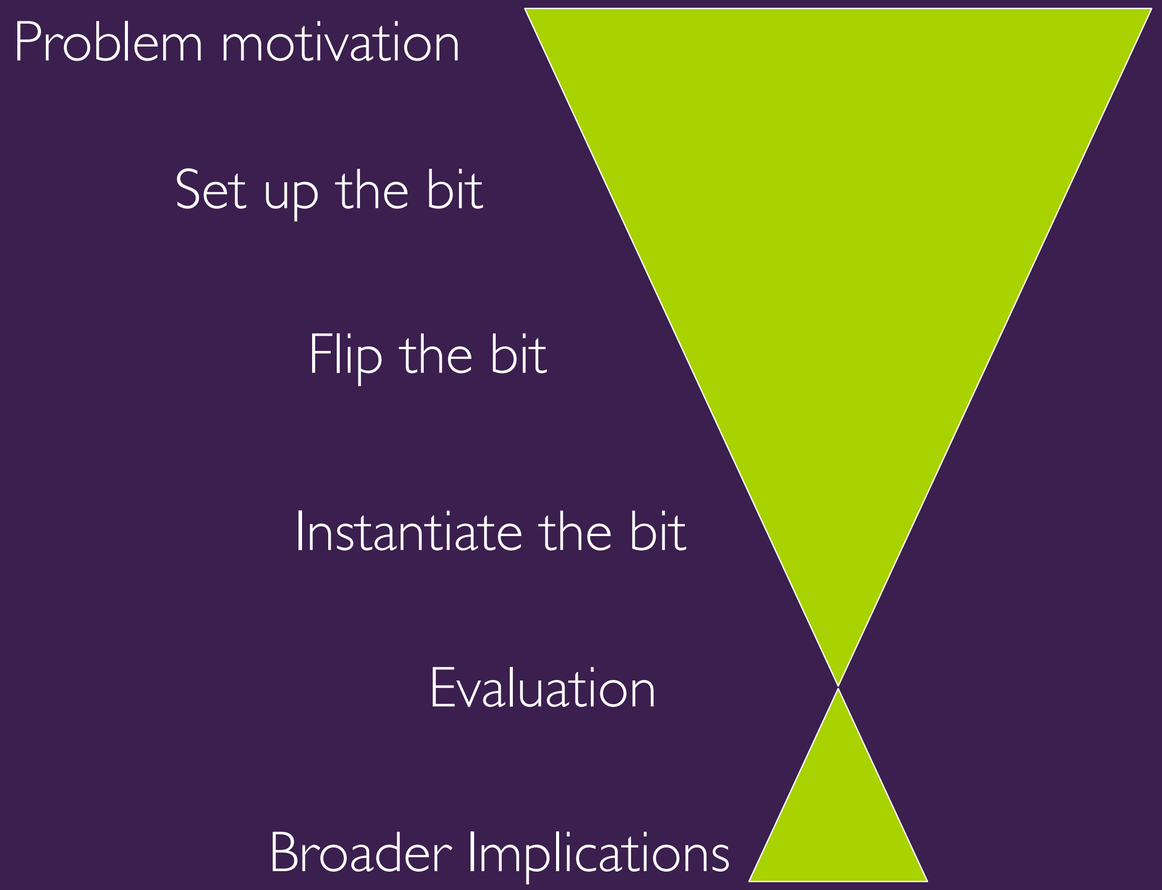
If you're right and the bit flip is how everyone should be approaching this problem from now on, what implications are there for the field?

This is your chance to stand on a small soapbox:

> Will it change the contexts in which we use this technology? Will it broaden usage?

But don't overplay your hand: it probably won't change all of computing.

Architecture of an intro



So in brief: use your literature search to motivate your problem and set up a bit.

Then, flip the bit and argue persuasively that this will address the problem. Explain how this solution gets built into your system or model.

Common Mistakes

Bit flip: Focus on low-level technical details instead of the **concept**

"We train with two images instead of one."

—— This is a technical detail. It does not explain what the bit flip is conceptually.

"We investigate the implications of incorporating temporal visual information. We use two temporally ordered images in the training process, as opposed to one static image."

—— It explains that the conceptual bit flip is adding "temporal" visual information. It also describes the instantiation: using two temporally ordered images instead of one.

Common Mistakes

Overstatements: Exaggerating the problem you have solved or the implications of your research. Expect that reviewers will be picky and will comb through each sentence. Strong claims need to be backed up by citations.

"Previous approaches to solving HPE and HAR together are deficient, data-hungry and neglectful of the potential of using natural language supervision to improve generalizability."

—— Do ALL previous approaches have ALL these problems?

"Previous approaches to solving HPE and HAR together were **limited** in their use of natural language supervision in learning a general model. "

Final Tips

Assume that a reviewer will comb through each sentence in your introduction and may challenge your claims.

Don't be vague, don't overstate.

Be precise, coherent, and convincing.

Don't worry, it takes a lot of practice even for PhD students!

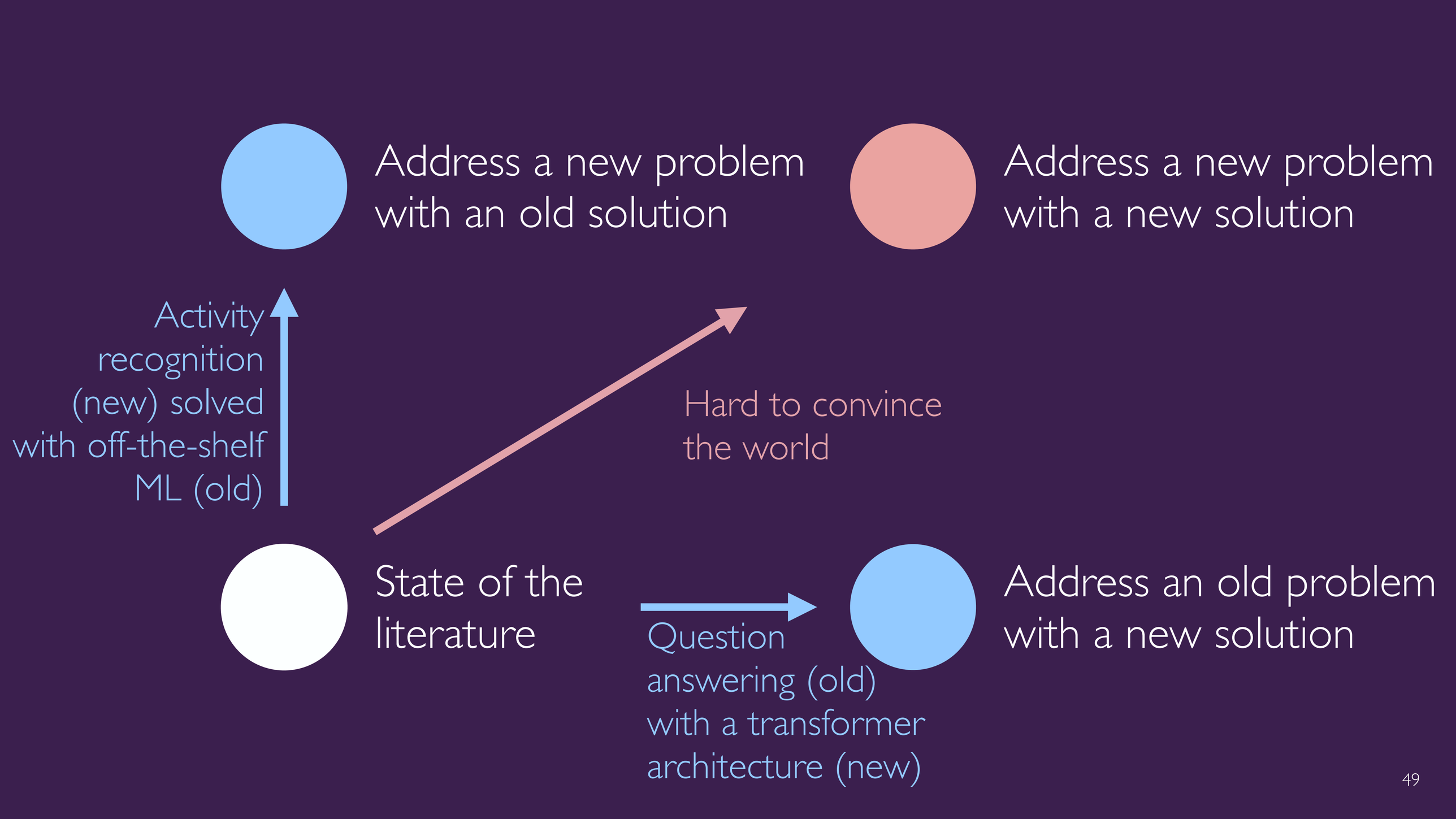
You will get to practice in A3. Your CA will give you feedback to improve upon for your final draft. If you improve upon the feedback, you will get a good grade.

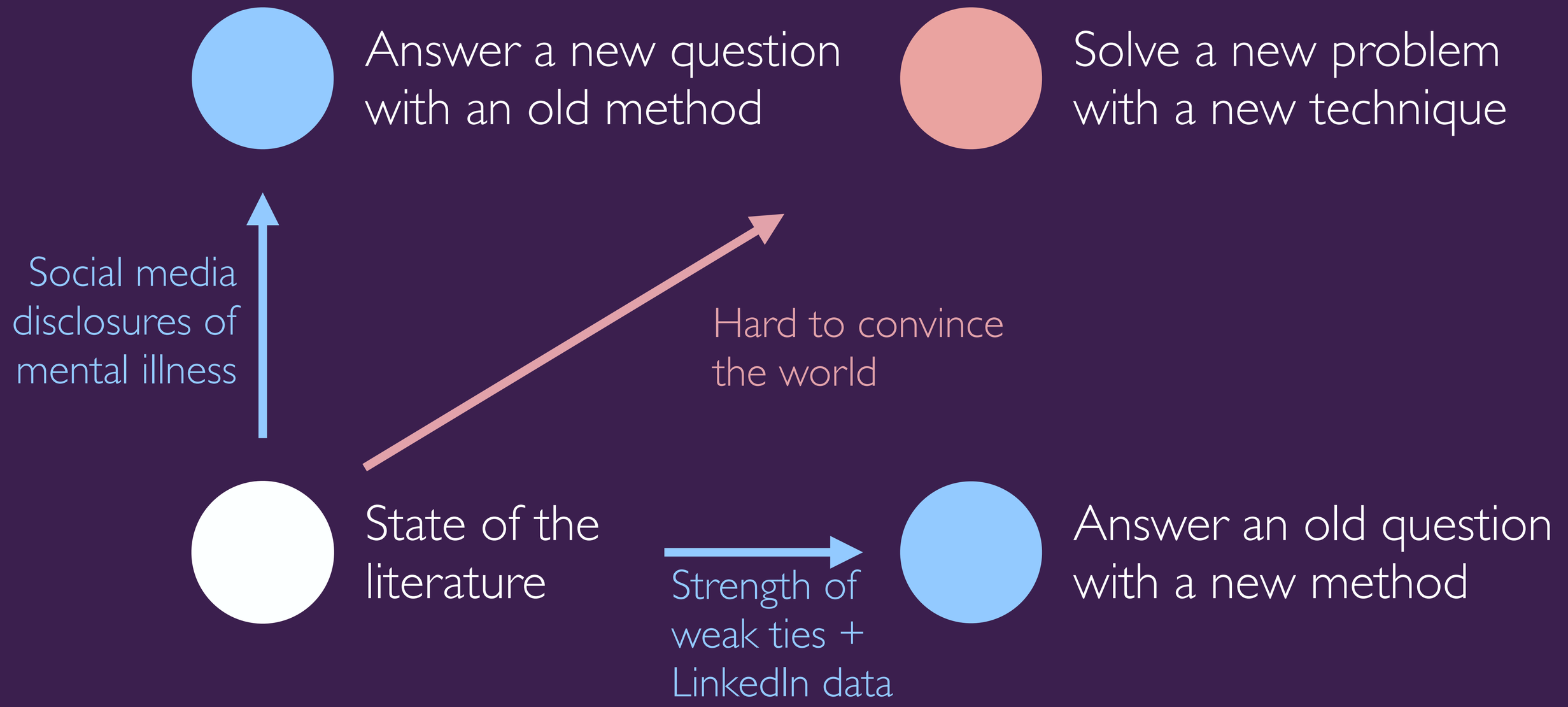
How to Write The Introduction

First, find your genre

There are a few different kinds of paper that are common:

- New problem / old solution
- Old problem / new solution


Address a new problem with an old solution
Address a new problem with a new solution
Activity recognition (new) solved with off-the-shelf ML (old)
Hard to convince the world
State of the literature
Question answering (old) with a transformer architecture (new)
Address an old problem with a new solution


Answer a new question with an old method
Solve a new problem with a new technique
Social media disclosures of mental illness
Hard to convince the world
State of the literature
Strength of weak ties + LinkedIn data
Answer an old question with a new method

Why only make one move?

When making an argument, you want to introduce one major new idea, to minimize the new ideas your listener needs to absorb. A research paper typically only flips **one** bit.

Typically you are spending the introduction making the case for your new idea. If you are trying to make the case for both a new problem and a new solution, a reader might disagree with either.

This is not to say that you can't do new problem / new solution; just that it's a risky varsity maneuver.

Use existing warrants

Certain ideas already have **warrants** in the literature: prior work already has proven their legitimacy. A warrant is a free pass!

- Old problem: the problem already has a warrant in the literature.
- Old solution: the solution already has a warrant in the literature.

From genre to intro

Old problem / new solution:

- Motivate the problem via prior work, which has already established the problem
- Set up the bit of how all prior work tried to solve it
- Flip the bit — your new solution
- Instantiate that new solution
- Implications

New problem / old solution:

- Motivate the problem via rhetoric, drawing on prior work making supporting claims
- Set up the bit: prior work is not equipped for this problem
- Flip the bit — how your approach draws on known ideas
- Instantiate that solution
- Implications

Start with an outline

Your idea should be fully understandable with only six sentences, a topic sentence per paragraph:

Problem motivation

Set up the bit

Flip the bit

Instantiate the bit flip

Evaluation

Implications

Keep it taut

Your goal is then to treat each outline point as a thesis sentence for the paragraph, and use the paragraph to prove that thesis. Don't stray and make other interesting but un-useful points.

Try it

Group up, and work on your outline [7min]

Share your outline, one sentence per topic, with another group in your section [1min each]

Assignment 3

Your group writes an Introduction to a paper for your project

- Outline the introduction
- Turn the outline into text
- 700-900 words

Details at cs197.stanford.edu and cs197c.stanford.edu

For 197: due Tues Apr 25, 10am

For 197C: due Tues May 2, 10am

Computer Science Research