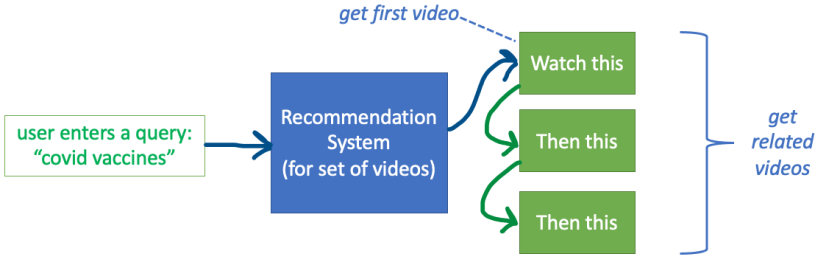
A Lecture on Identifying Social Threats in Data Structures and Algorithms

Kathi Fisler, Brown University Dept of Computer Science

Context: *This is a lecture from a second-semester course (at Brown University) on object-oriented programming, data structures, and algorithms. This lecture occurs just over halfway into the semester-length course. The course has already covered decision trees and hashmaps (both conceptually and with homework assignments that use and implement them). The course has also covered the model-view-controller software architecture, and used that to decompose systems into classes. The students have had a couple of assignments that engage them in thinking about socially-responsible computing, both in this course and in their first semester course. This lecture is where we first discuss how to systematically look for social threats in the kinds of projects and applications that would normally be covered in the course (and several other courses that follow this one, as part of the dept-wide Socially-Responsible Computing program in Brown CS).*

1 A Data Structure Selection Exercise

Imagine that you've been asked to design an application to recommend videos to users based on initial search terms. A user would enter a search query, and the system would generate a sequence of videos to watch based on the query. The system would show the videos in order until the user exited the application.



In terms of data, the application has the set of videos (with information on title, actors, production year, genre, etc), as well as some information on which videos have been watched by which users.

Your team has two proposals for organizing the video collection for purposes of answering queries:

- Make a decision tree with the videos at the leaves
- Create a hashmap/dictionary from query terms to sets of matching videos

Stop and Think: Sketch out how you might use each of these proposals. Draw an example of each proposed data structure and briefly describe the algorithm you will use to get the first video and the sequence of related videos to show after the first one.

Stop and Think: Based on what you've sketched out, which data structure do you prefer and why?

Regardless of data structure chosen, the most common response to "how to generate the sequence" involves navigating either the decision tree or the hashmap to get a set of videos, then generating a sequence

leveraging the user profile data. The data structure choices usually boil down to which seems easier to use for this task. There are, of course, other approaches, but let's focus on this common one for now.

Instructor Note: *We do this as an interactive exercise, with students working on the two questions in small groups, then class-wide reporting of ideas and tradeoffs.*

2 Social Threats in Video Recommendation Algorithms

In 2019, danah boyd, a technology and social media researcher at Microsoft, Data & Society, and NYU, gave a talk called "The Fragmentation of Truth". The video and a transcript are available online. Her talk focused on a series of behaviors of the YouTube recommendation algorithm that enable some users to control what content other users get to see. Her points include:

- If people search for content that doesn't exist, YouTube waits for users to upload something to the platform (rather than see what other quick-response sites like Twitter might be reporting).
- People find content by searching on keywords. If the search algorithm uses simple keyword match, a group of people can agree to use keywords/phrases to promote content. Then if journalists or influencers promote those phrases, traffic can get directed to the videos selected by the group.
- Searching on outdated terms returns outdated results, devoid of context (including newer thinking on the terms)
- If content largely takes a one-sided opinion, all search results reinforce that opinion

The design opens an opportunity for a group to control what others see: identify a "hot topic" search term with few videos, upload content tagged with that term, get friends to watch and like the uploaded content, and end up with a single viewpoint promoted by a coordinated group of people. Her article goes into more detail of this real-world scenario.

This scenario is a *social threat*: a negative societal consequence of an application or algorithm. Think back to the original data structure proposals that you considered: a naïve hashmap design of the data structure enables this negative scenario. The whole benefit of the hashmap is that it lets someone quickly narrow in on a set of related videos. But that design is insufficient to then prevent the negative scenario, if all of the recommendations stay in that narrow corner of the hashmap!

The key point here is that adverse consequences of systems can arise from early-stage design decisions about data structures and algorithms. A responsible designer needs to weigh the potential for social threats like this *during the design phase*. If issues like this are only discovered during pre-release testing or once the system is in the field, it will be too expensive to redesign the data structures and algorithms to avoid the problem cleanly.

The main goal of this lecture is to get you started thinking about *how* to identify social threats in systems that make use of data, data structures, and algorithms.

3 Another Exercise: What is Fairness?

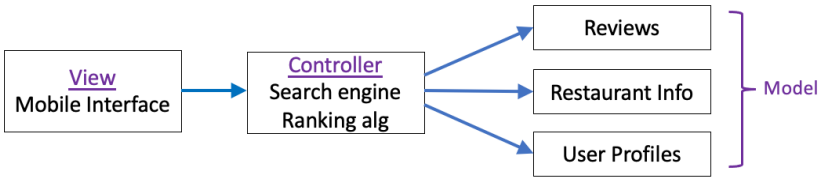
The following list contains a sample of responses from first-year university students on what it would mean for a restaurant-rating app like Yelp or Dazhong Dianping (an analogous site in China) to be fair.

Instructor Note: *My course asks this question on a background survey at the start of the course. I present results of that survey with quotes from actual/current students at this point.*

1. Take into account that lower-income neighborhoods and cities would really benefit from having positive reviews about local restaurants. Platforms should make an effort to focus on advertising those local places, in addition to [...] all the top-rated restaurants of any given area.

2. Instead of allowing the user to choose a holistic rating (scale from 1 to 5 stars), the system should ask a series of questions and calculate the corresponding rating accordingly.
3. A fair system would not reflect the favorite restaurants of the programmers, for example.
4. A good system would recommend a number of different spots from different categories based on things consumers value, like highest average reviews, best customer service, up-and-comings, etc.
5. The main issue with review websites is a lack of empathy. The people who write these reviews often forget that there are actual people who cook the food, actual people who serve the food and actual people who own the restaurant.
6. The system should have a way of matching the review with the person who would've been responsible for the dissatisfaction the customer is facing. [...] Furthermore, [...] websites should work with these restaurants to have a unique number given to every customer in the bill after the meal. Only through using this number shall customers be allowed to write reviews on these websites.
7. All these websites depend mainly on ad revenue, which means that inherently, sponsored content that appears above the top results are already biased for the websites to run the service.
8. Fair systems would probably allocate ratings and reviews based on actual current service, and also maybe compared to other restaurants around the area (so that high end hotels or restaurants aren't compared to regional places)
9. The system is fair if it ensures that 1) the people who rate a restaurant have actually eaten at the restaurant and 2) people who review a restaurant are customers and not restaurant owners.
10. A diverse body of individuals evaluated the restaurant, at different hours/mealtimes, and many reviews, so that all aspects of the restaurant are reflected
11. A fair system would deny restaurant owners the ability to remove or edit removes. Restaurant owners would have the opportunity to dispute negative reviews on the forum or through the website administration. Customers would need to register an account to establish uniqueness and prevent spam or duplicate reviews. This would also prevent store owners from creating fake positive reviews.
12. There should be support for a variety of languages [...] the main themes that should be upheld are transparency and accessibility.

Here is a high-level model-view-controller architecture diagram for such an application:



Stop and Think: For each response in the list, identify who is potentially harmed by this unfairness. Then, consider which aspects of the application (the model, view, controller, or other) plays a role in enabling that unfairness and how.

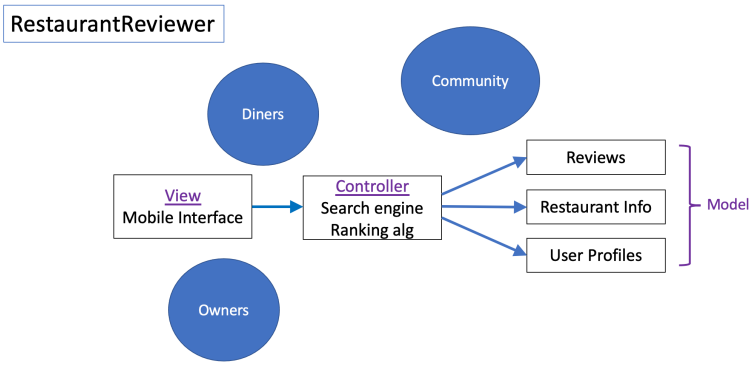
4 Systematic Analysis of Design Threats

4.1 Impacts Start with Stakeholders

One goal of the previous exercise was to have you recognize that there are many stakeholders who contribute or who can be influenced by an application. In restaurant review apps, for example, there are

potential impacts on each of diners, restaurant owners, restaurant employees, owners of other restaurants, and surrounding neighborhoods.

Taking a systematic look at social impacts starts from trying to identify the stakeholders who *influence* or *are influenced by* the application or algorithm. Notice that our original model-view-controller diagram does not include stakeholders, even though they interact with the system and its data in significant ways. Therefore, one place to start is by adding the stakeholders to the MVC diagram for a program. The following diagram has added some of the stakeholders in blue circles.

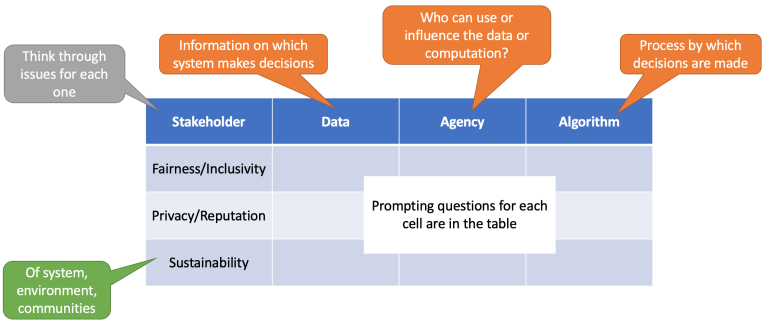


4.2 Impacts Fall Into General Categories

Another goal of the exercise was to have you generate ideas for possible social concerns or threats. As you did this (and as we reported out across the class), it probably felt like there was a big pile of possible problems. But if we step back, we start to realize that these threats can be clustered into several larger categories, such as fairness, inclusivity, privacy, reputations, and sustainability. Having categories opens possibilities for us to look for threats systematically.

4.3 Identifying Threats Systematically

Combining stakeholders, threat categories, and system architecture, a process for identifying threats emerges. We can view the process of looking for social threats as filling in a table (once per stakeholder) that puts technical aspects of the system on one axis and categories of social concerns on the other:



Roughly, the *data* component relates to the model and data parts of the system, the *algorithm* component relates to the controller and data parts of the system, and the *agency* component relates to the stakeholders and view parts of the system.

This figure shows only an outline of the table to give you the core idea. A more detailed table, with specific prompting questions within each cell, is available at this URL (table on page 2, some framing work about stakeholders and other considerations is on page 1):

Stop and Think: Return to the Recommender system design activity from the start of the lecture. Pick two cells of the table, think through the prompting questions, and see what potential threats (whether boyd's or your own) land in different cells of the table.

5 Key Takeaways

The goal of this lecture was to start showing you how to think systematically about social threats to systems. You have been learning how to evaluate systems for run-time, space usage, and performance. This is another aspect of the same goal: evaluating system designs for social impacts (which could be positive or negative).

At this point, we expect that you could work from the detailed table and start to identify questions to consider, at design time, about a proposed project. We haven't taught you how to mitigate the problems that you identify (that will happen in later courses and indeed throughout your professional career). But we do want you to start building the instinct to include social-impact analysis along with the more quantitative forms that you have learned so far.