CS 5342

Trust and Safety: Platforms, Policies, Products

Fall 2025

Assignment 3: Automated moderator (Bluesky Labeler)

Overview

In this assignment, you will gain first-hand experience in designing and evaluating an automated moderator, that is compatible with Bluesky's customizable approach to moderation, to handle a harm type of your choice. You are encouraged to implement the policy proposal you outlined in Assignment 2 because you'll have spent significant time grappling with it, but you are also free to choose a different problem to tackle if you don't think you can implement your solutions from Assignment 2. Your choice to build on Assignment 2 or start anew will not affect your grade.

We expect you to comprehensively test your code for this component. The extent to which your code and testing is well-documented will constitute a portion of your grade. This part of the assignment is more open-ended, so you'll have to demonstrate to us that you've thought through how you can verify that your implementation will meet your stated moderation goal. The creativity you demonstrate in your chosen problem/solution will also constitute a portion of your grade.

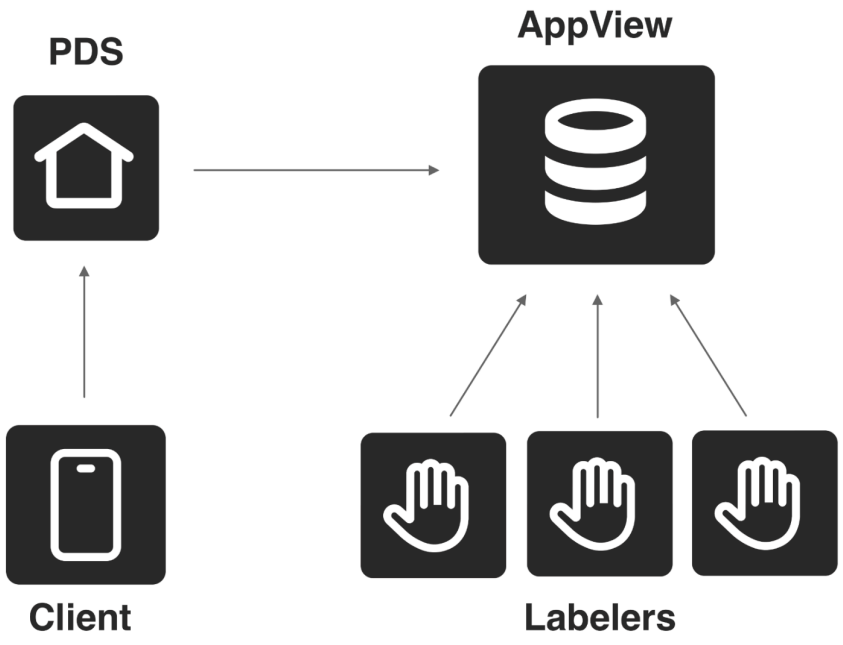
Bluesky moderation infrastructure

Bluesky's moderation system is decentralized, designed to give users more agency over how content is filtered, organized, and displayed. A key part of this system is the use of *labelers*—services that generate labels on posts and accounts. Users who subscribe to labelers can configure how these labels are applied to the posts they see. For instance, your service may attach a label for spam or NSFW content (throughout, *content* will refer to both posts and accounts). Some users may wish to hide such content altogether, others may prefer that a badge be attached to it. Recall that content moderation is not solely about blocking harmful content. It can also be about organizing and displaying content in a way that is helpful to users. Similarly, labelers are not just for marking definitely objectionable posts and accounts. Here are a few examples:

- The pronouns labeler allows users to display a badge on their profile indicating their pronouns that subscribers to the labeler can see.
- The US Government Contributions labeler will apply badges to the accounts of representatives with the organizations that fund them. After subscribing to this labeler, you can look up Alexandria Ocasio-Cortez's account, and see badges indicating that her donor list includes employees of, or PACs tied to, Alphabet and the City of New York.
- This popular labeler attempts to identify AI-generated imagery.

You can find more examples of labelers at Bluesky-labelers.io. We encourage you to try some of them out before starting the assignment.

You can find a detailed discussion of the Bluesky moderation infrastructure here. We provide a high-level overview below. Account data is hosted at a *personal data server* (PDS). This data is distributed, via a *relay*, to an *AppView* service. Labelers are services that generate labels on posts and accounts. These labels are sent to the AppView. When user client devices download posts from the AppView, they obtain labels associated with the content they received, depending on what labelers they are subscribed to.



The figure above[1], from the Bluesky moderation infrastructure overview, provides a visual representation of how labels are generated and sent to the AppView. Bluesky provides its own labeler that handles platform-level moderation policies. In addition to this, users can opt into

[1] https://docs.bsky.app/blog/blueskys-moderation-architecture

other third-party labelers for additional layers of moderation. In an actual production environment, your labeler would likely ingest posts from the firehose, which provides a stream of content as it is disseminated through the network. **However, for the purposes of this assignment, your labeler will be ingesting posts from a CSV file.** This will allow for easier testing and debugging.

To learn more about bluesky's moderation, here are some other resources:

- AT Protocol spec
- AT Protocol Python SDK documentation
- Bluesky developer discord
- List of Bluesky labelers
- Broader developer docs for Bluesky

Set-Up

Make sure you have nodejs and python installed, and you may also need to install a code-editor (e.g. visual studio code)

Part 0: Optional & ungraded coding exercise

Below is a recommended and ungraded exercise to help you get comfortable coding content moderation systems. You can check your work using the autograder file posted on canvas.

Starter code

You can access the starter code from last year's class Github, under the `bluesky-assign3` directory. Open up `automated-labeler.py`. You'll notice that we provide a constructor for your labeler and a `moderate_post` function. This function takes as input a url to a Bluesky post and produces an `List[str]`, i.e., the function returns a list of string, corresponding to a label if there is one to be added for the post, or a `[]` value if there are no labels to add. When you run your labeler via `test-labeler.py`, the output of `moderate_post` will be used to emit a label via the `label_post` function defined in `label.py`. You can configure whether to actually emit labels to the Bluesky network via a command-line argument for `test-labeler.py` – while you're testing your code, you shouldn't be emitting labels. You should decompose the logic in `moderate_post` across different functions that you'll define for your `AutomatedLabeler`.

Label posts with T&S words and domains

A common moderation technique that platforms employ is text matching against a list of known harmful text. In this part of the assignment, you will implement this technique to label posts

containing Trust-and-Safety-related words/domains. We sourced the words from the TSPA glossary. You will find this list in `t-and-s-words.csv` and a list of T&S domains in `t-and-s-domains.csv`. For each post in `input-posts-t-and-s.csv`, apply a “t-and-s” label to those that match either list. Make sure to take into account case sensitivity. If the word “moderate” is on the list, then a post containing the word “mOderAtE” should also be labeled.

Cite your sources

Label posts that link to news articles with the news publications with which they are affiliated. You will have to create labels for the following publications: CNN, BBC, NYT, Washington Post, Fox News, Reuters, NPR, AP. The file `news-domains.csv` will contain a list of the domains you should scan for, along with the label to apply. For each post in `input-posts-cite.csv`, apply the appropriate label(s). Note that if there are multiple news links from different sources, then multiple labels should be generated for each source. If there are multiple links from the same source, then only one label should be generated.

Dog labeler

Many platforms employ a technique called *perceptual hash matching* in order to detect harmful or illegal images. An image is passed through a perceptual hash function, which outputs a bitstring (a sequence of 0’s and 1’s), such that two similar images should hash to similar bitstrings. In this part of the assignment, you will use this technique to identify pictures of dogs that match a known list (the dog-list).

For posts in `input-posts-dogs.csv` containing an image matching the image dog-list (the images in the `dog-list-images` directory – sourced from WeRateDogs), apply the “dog” label. A match is defined as the image being within a hamming distance of `THRESH`[2] of the target image’s perceptual hash. You can use the PHash implementation provided here to perform perceptual hashing.

We leave it to you to figure out how to extract the image(s) contained within a post. You’ll find it helpful to consult the `atproto` documentation along with the `PIL` and `requests` python modules. See if you can notice a pattern in the URLs associated with post images.

[2] This is a constant that will be provided in the starter code.

Part 1: Designing & implementing your policy proposal

We expect your problem selection and solution to demonstrate a reasonable level of creativity, sophistication, and involvement. For instance, tackling toxicity by making a call to the Perspective API for each post and attaching a “toxic” label if it exceeds some threshold would not suffice. You will likely have to iterate on your policy proposal and implementation to achieve something reasonable. Document this process and discuss it in your presentation.

You should begin by gathering data on the harm you plan to tackle. This will inform your testing approach and solution design. You can use the ATProto SDK to crawl Bluesky and filter for content that may be relevant to the harm you address. You can also leverage research done for Assignment 2. Part of your grade will be based on the description, execution, and efficacy of your testing setup. Depending on your approach, you may have to manually label some of the data you collect. **We do not want you to deal with illegal or severely harmful content (e.g., sale/solicitation of illegal substances, CSAM, etc).**

Remember that precision in labeling at scale is difficult – and you only have a few weeks. For that reason, we encourage you to choose a labeling implementation that recognizes it is detecting *potentially sensitive* content rather than one that is categorical about finding the harmful material – unless you can be highly confident about the accuracy of your endeavors. You can help yourself by being precise about what you call the labeler; you can then explain why that labeler might help fight the harm you care about in your presentation / video. Here are some illustrative examples:

- “Potentially soliciting financial information” is a better labeler than “Fraud posts”. You might use a combination of text-matching and LLM reasoning to label posts that include certain brand names tied to money exchange (e.g., Venmo, CashApp) *and* a call to action (e.g., “Send me your” or “Give me”). Recognize that people often provide this information during emergencies, for fundraising, or as tips for their online work (e.g., on Patreon).
- “Addresses content that has been fact-checked before” is a better labeler than “Fake news” both because it is more precise and because you are not making a definitive judgment about the veracity of the Bluesky post, which will be very hard. To build such a labeler you may choose to lean on the Fact Check Explorer API or the open source Community Notes data.

For inspiration, we provide below a non-exhaustive list of inputs, signals, and tools you may consider using in your labeler:

- Perspective API for toxicity scoring
- Google fact checking tools, other fact checking databases/APIs
- Analysis of the Bluesky network – looking at follower lists, number of posts/replies etc. other metadata. This could be helpful for analyzing particular communities.

- User input – users can message your labeler/react to posts. This can inform a collaborative voting/labeling approach
- Non-profit, human rights, and/or legal groups that have categorized organizations in ways that may be useful for labeling purposes (e.g., RSF Freedom of the Press index)
- LLMs (open-sourced models like Llama), computer vision models

Throughout this class we have discussed how safety measures can in turn be abused. We encourage you to continuously check the work that you are doing for unintended consequences and follow our course's policy on engaging with harmful content. Consider the ethical implications if you were to deploy your labeler on bluesky. . For instance, you could see how labeling non-notable individuals for their perceived political position on a hot-button topic could lead to doxxing campaigns or worse. Discuss how you thought about these ethical considerations in the video presentation.

Your implementation should be in a file named `policy_proposal_labeler.py` and the dataset you plan to use should be in a file named `data.csv`.

Part 2: Testing and Evaluation

We expect you to test on a reasonably large number of posts (e.g., somewhere in the ballpark of ~150 posts). Your test posts can be from real posts online or synthetic posts. For example, you can use Bluesky's API to find posts (e.g. app.bsky.feed.getPosts). However, it's important to be strategic on the makeup of posts you choose from. Select posts to stress-test your labeler, think of edge cases that could "break" it, and ensure you have a diverse set of posts to test how holistic your policy is. If you are building a classification model that decides whether a post violates a policy or not, or what kind of harm category it falls into, you need **ground truth labels,** this means you need to source what is true from somewhere. This could be from other public datasets or as a team you may need to annotate it yourself. If you do annotate it, it's important to document how you did it (e.g. rubric for the annotators and if 100% consensus is needed).

Once you have your posts, evaluate the accuracy, precision, and recall of your labeler (for labelers highly dependent on user input, this analysis may look slightly different). You should also discuss the efficiency and performance of your labeler in terms of the amount of computation and memory it requires – these are things you may consider measuring, e.g., How long does it take your labeler to make a decision on a particular post? How much memory does it consume? How much network communication does it require? Your team should also analyze the results, explaining reasonings as to why your code performed as it did and present concrete next steps including how to improve the labeler and future tests.

It's important to note that you select an evaluation method that makes sense given your application. For example, if you build a classifier then accuracy, precision, and recall are metrics

you'd like to capture. However if you have a non-classification task of a user-facing application, you may want to emphasize usability, scalability, and efficiency.

Importantly, your team's code will be evaluated, ensure that you provide detailed instructions on how to run and test your code and ensure that your code runs (without errors) to where we can reproduce your results.

Deliverables

We expect you to submit the following deliverables on Gradescope:

- A well-documented implementation of your labeler in python
    - A readme file with your group number and group names, list and description of all the files you've submitted, and how to run your test, these detailed instructions here are not part of the 2-page report detailed below.
    - Submit all files including `policy_proposal_labeler.py` and `data.csv`
- A **two-page PDF report,** summarizing: (1) motivation, (2) summary of your approach / policy, (3) description of your test; (4) the test results (e.g., false positives, false negatives, etc.); and (5) analysis that discusses the results, next steps, and what they imply about your use case.
- A **10-minute video presentation** describing your implementation choices, your testing approach, running your tests and showing the results (e.g., via screencapture), and an evaluation of your solution for addressing the chosen harm.
    - Your presentation slides and any other materials you used to create your video

Video Presentation

- ~10 minute recorded video, each team member must speak in the video presentation
- Introduce, motivate, and explain the harm you aim to mitigate, along with your proposed policy
- Discuss the various approaches you tried out, explain what hurdles where and what you needed to go back on in your policy to make a better implementation that reflects it
- Give a high-level technical overview of your implementation
- Discuss your approach to testing and evaluation
- Provide a demo of your labeler in action; this means running your test in real-time and showing the results
- Analyze the ethical implications of deploying your labeler
- Talk about future areas for improvement

Grading

Your grade in the assignment will be made up following components:

Component	Weight	An excellent assignment will…
Part 1: Efficacy of policy proposal implementation	45%	● Introduce, motivate, and explain the harm you aim to mitigate, along with your proposed policy ● Provide functional and working code that properly implements the articulated policy proposal ● Discuss the challenges in implementing the proposal and document your iteration on the code and policy ● Provide a demonstration of the labeler ● Provide a technical overview of your implementation ● Discuss the ethical implications of deploying the labeler
Part 2: Description of testing and evaluation	45%	● Describe the process of collecting and labeling data ● Document testing set-up and design ● Argue for the appropriateness of this testing design ● Provide empirical results for your evaluation for accuracy, precision, and recall, as appropriate (if your solution is classification-oriented; indicate you have stress-tested the code and feature for an at-scale implementation if not, for example by attempting to emit a certain number of labels ~150) ● Analyze and discuss the performance of the code ● Discuss future areas for improvement
Video Presentation	5%	● Create a compelling video presentation that tells a story with a clear motivation on the harm, your iterative process of implementing an effective policy, your tests and evaluation, and future areas for improvement with the goal of deploying the labeler ● A demo of running the labeler and tests in real-time ● Each team member must present during the video presentation ● Well-designed slide presentation
Code documentation and style	5%	● Provide legible, well-formatted, well-commented code ● Use meaningful variable names ● Decompose complex logic into smaller helper functions ● Include well-written documentation that enables other developers to use and modify the code