CS 5342

Trust and Safety: Platforms, Policies, Products

Spring 2025

Assignment 3: Automated moderator (Bluesky Labeler)

Overview

In this assignment, you will gain first-hand experience with Bluesky's customizable approach to moderation. We'll walk you through implementing a labeler, which is a service that attaches categorical labels to Bluesky posts and accounts. Users who subscribe to your labeler 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.

Your task

In the first part of the assignment, you will build an automated labeler that will apply labels to Bluesky posts based on their text content. We will provide a test set of posts and their expected labels in a CSV file. You should not hard-code these labels in your implementation – we will test your code on some examples that do not appear in this test set, and a portion of your grade will be based on your labeler's accuracy on these instances. **Furthermore, if you do hard-code**

labels for particular posts, you will receive a 0 for the functionality score. If your labeler produces nothing for all inputs, it will also receive a 0.

In the second part of this assignment, you will implement your own automated moderation policy as a Bluesky labeler. This can be the policy you articulated in Assignment 2, but you are free to choose another topic. 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.

Ethical guidelines

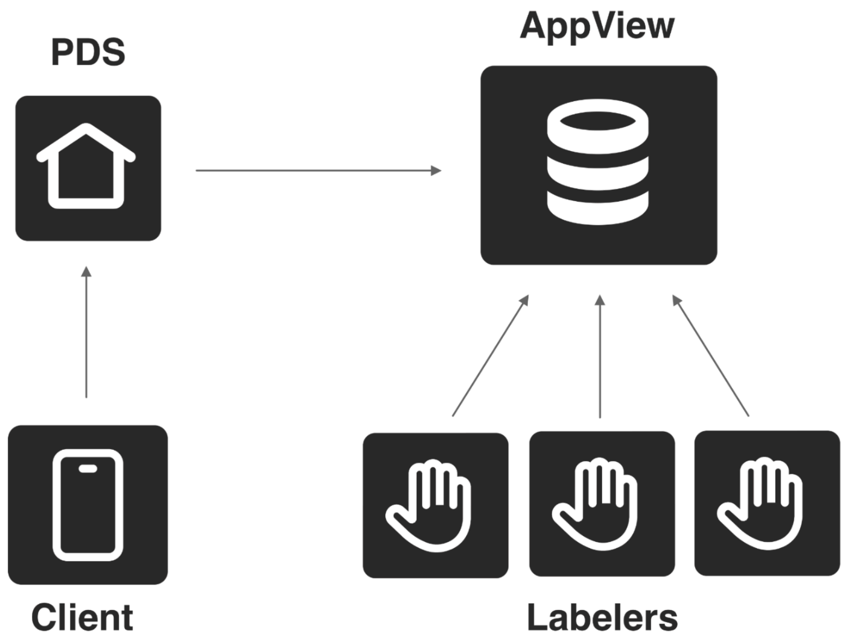
Throughout this class we have discussed how safety measures can in turn be abused. We encourage you to continuously check the work that you are doing for unintended consequences and follow our course's policy on engaging with harmful content. You should be particularly careful when completing Milestone 5. This will include documenting your process carefully, clearly signposting the exercise as an academic effort, and providing a way for labeled users to express any concerns with the label.

Deliverables

- A well-documented implementation of your labeler in python
  - Your Part I implementation should be in `automated_labeler.py`
  - For Part II, create a file named `policy_proposal_labeler.py` – you can base that implementation off of the code we provide for you.
- A 10-minute video presentation describing your implementation choices, your testing approach, and an evaluation of your solution for addressing the chosen harm
- Your presentation slides and any other materials you used to create your video

Bluesky moderation infrastructure

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 other third-party labelers for additional layers of moderation. You will implement and run one such labeler for this assignment.

Starter code

You can access the starter code from the class [Github](), under the `bluesky-assign3` directory.

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.

In this assignment, your labeler consists of two components: the first is the *labeling server*, which interfaces with the AppView to attach labels to content. This is a Javascript program that uses the [skyware/labeler]() library. The second component is your *labeler bot*, which you will implement as a python program that will interface with the labeler server to produce labels.

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

Milestone 1: Labeler Setup

In order to create a labeler, you need a public domain to host the labeling server and a Bluesky account associated with the labeler. We will handle the hosting of the labeling server for you. If you're interested in an additional challenge and want to own/operate your own labeling infrastructure, you can consult this guide. For the purposes of completing this assignment, you are not required to make your labeler live (i.e., emit public labels). In fact, for Part II, you should be careful to check with us before you start attaching labels to public posts. However, we give you the option to emit labels so you can see your hard work in action on the Bluesky network.

Start by creating your own GitHub repository with a clone of the starter code.

Install the necessary software

Make sure you have nodejs and python installed. Install the skyware labeler package with the following command:

```
npm install @skyware/labeler
```

Make sure you have Python 3 installed, along with the ATProto, Dotenv, Requests, and Perception modules:

```
pip install atproto dotenv requests perception
```

Testing that you can access posts

Run the following command in the starter code directory to ensure that you can access posts.

```
python get_post_test.py
```

If this runs successfully, you are ready to begin the assignment. Testing that you can emit public labels (the following section) is optional.

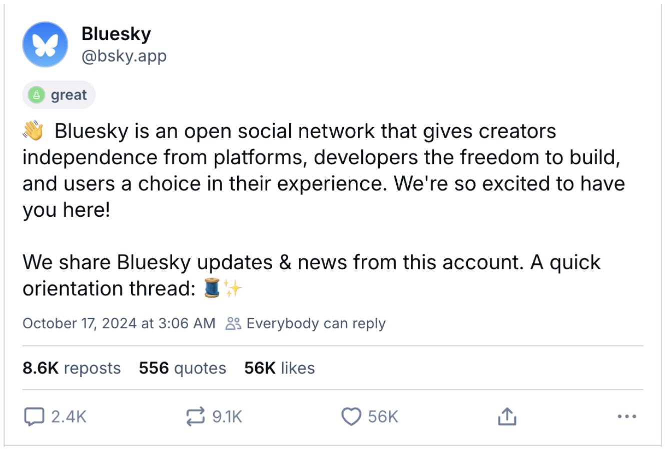
Testing that you can emit public labels

From your browser, visit https://<your-domain>/xrpc/com.atproto.label.queryLabels
This will display all the labelers that have been issued by your labeler. Initially, this list will be empty. Let's change that. Run the following command to apply a label to the bsky.app account:

```
python label.py post https://bsky.app/profile/bsky.app/post/3l6oveex3ii2l great
```

This applies a "great" label to the post at the specified URL. You can use the skyware/labeler command line utility to modify the labels that your labeler supports.

Subscribe to your labeler from a different account (perhaps your personal bluesky account) and visit the post in the URL to observe that the label has been applied:



Bluesky
@bsky.app

great

👋 Bluesky is an open social network that gives creators independence from platforms, developers the freedom to build, and users a choice in their experience. We're so excited to have you here!

We share Bluesky updates & news from this account. A quick orientation thread: 🧵✨

October 17, 2024 at 3:06 AM Everybody can reply

8.6K reposts 556 quotes 56K likes

You can also use the Label Scanner tool to verify that your label was applied to the post.

Structure of the starter code

At this point, you have confirmed that you can emit labels for particular posts and accounts via the command line tool. That's a great achievement already – you have the essential infrastructure for running your own third-party moderation service on Bluesky, congratulations!

Now, you'll automatically apply labels based on posts that meet certain criteria. 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 `Optional[str]`, i.e., the function returns a `str`, corresponding to a label if there is one to be added for the post, or a `None` value. You **must** implement your labeling logic in this function as this will interface with the auto-grader for the assignment. 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.

A portion of your grade will consist of your coding style – your code should be legible and well-organized. You should decompose the logic in `moderate_post` across different functions that you'll define for your `AutomatedLabeler`.

For Part I, we will provide an isolated testing script for you to test how your code generates labels. This will be the same script that our auto-grader will use in determining the functionality score for Part I. You may find this script helpful for your testing set-up in Part II as well.

[Part I] Milestone 2: 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.

[Part I] Milestone 3: 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.

[Part I] Milestone 4: 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 II] Milestone 5: Implementing your policy proposal

In Part I of this assignment, you gained familiarity with the AT protocol and implemented automated moderation routines. Using those skills, you'll extend your labeler 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. Recall that your implementation for Part II should be in a file named `policy_proposal_labeler.py`.

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, computer vision models

For full transparency, you should **make it clear in the description of your labeler account that your labeler is part of an educational exercise and that it should not be trusted for complete accuracy.** You should also collect and respond to any serious criticism.

Take care to consider the ethical implications of deploying your labeler and ensure that it does not lead to or provide a vector for abuse/harm. 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. Please feel free to reach out to us if you want to gut-check your proposed labeler for possible harmful consequences.

Presentation guidelines

- ~10 minute recorded video
- 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
- Provide a demo of your labeler in action
- Discuss your approach to testing and evaluation
- Analyze the ethical implications of deploying your labeler
- Talk about future areas for improvement

Testing and evaluation

We expect you to test on a reasonably large number of posts (e.g. somewhere in the ballpark of ~100), and 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?

Resources

As you build an application that interfaces with Bluesky and the AT protocol, you'll likely have conceptual questions about how the protocol and python SDK work. Additionally you may wonder about useful APIs for implementing your Part II solution. We list relevant resources below:

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

Grading

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

Component	**Weight**	**An excellent assignment will…**
Part I Functionality	35%	● Accurately generate labels for posts as specified in the exercises for Part I
Part II: Efficacy of policy proposal implementation	30%	● Introduce, motivate, and explain the harm you aim to mitigate, along with your proposed policy ● Provide functional 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 in action ● Provide a technical overview of your implementation ● Discuss the ethical implications of deploying the labeler ● Discuss future areas for improvement
Part II: Description of testing and evaluation	30%	● 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 accuracy, precision, and recall, as appropriate (if your solution is classification-oriented) ● Analyze and discuss the performance of the code
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