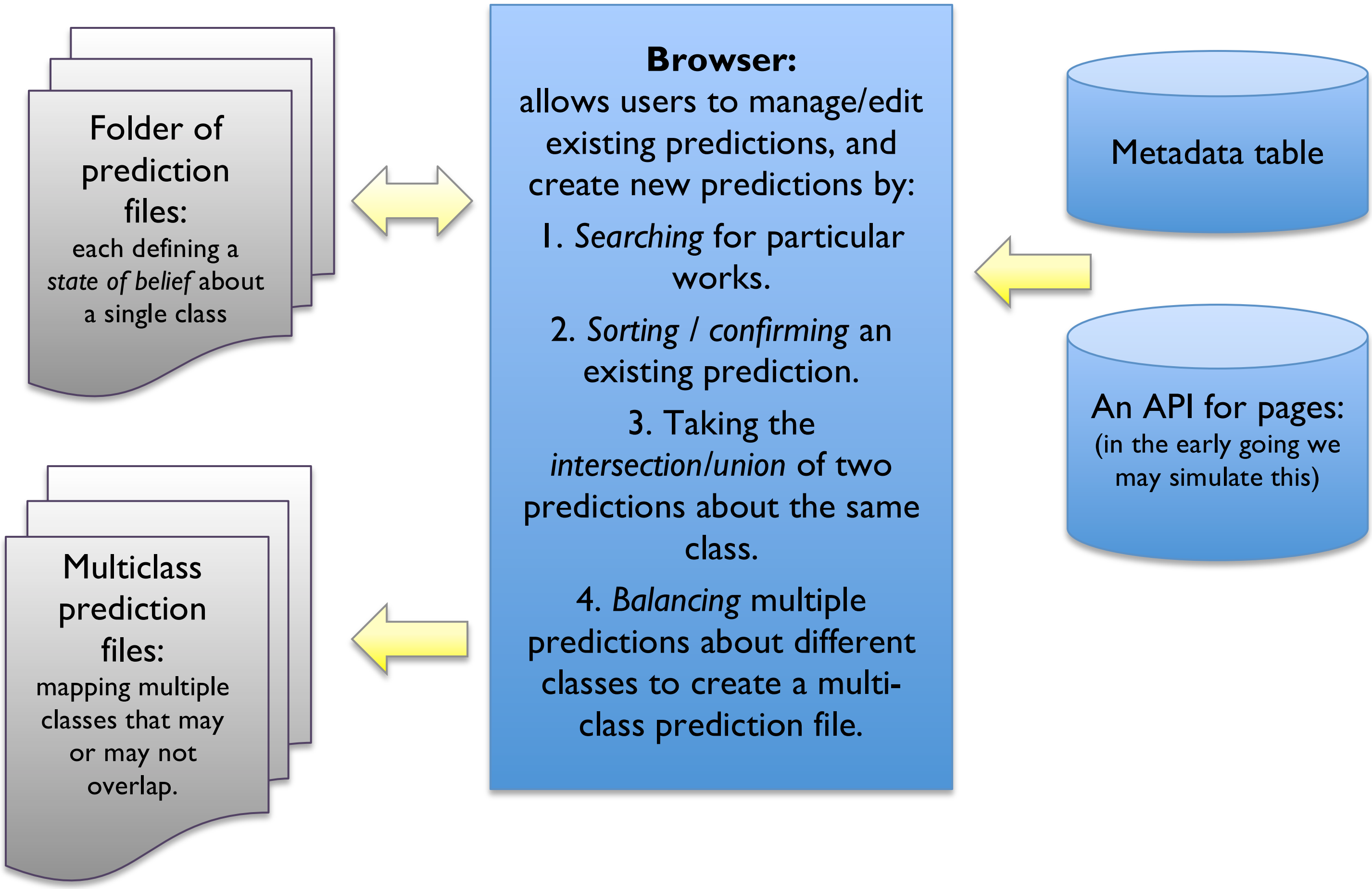
The Metadata Browser

I'm not sure how much of this we can get done this summer; I've actually got another year to build the whole thing. But for what it's worth, here's an overview of the plan.



Folder of prediction files: each defining a *state of belief* about a single class

Multiclass prediction files: mapping multiple classes that may or may not overlap.

Browser: allows users to manage/edit existing predictions, and create new predictions by:

1. *Searching* for particular works.
2. *Sorting / confirming* an existing prediction.
3. Taking the *intersection/union* of two predictions about the same class.
4. *Balancing* multiple predictions about different classes to create a multi-class prediction file.

Metadata table

An API for pages: (in the early going we may simulate this)

To think about this from a user-experience angle: the central user interface is probably a window with a list of data sources presently available within a particular collection. Your options would include prediction files, as well as the option of working with the collection as a whole. Before you can *do* anything (search, sort, confirm, etc), you'll need to select one of these data sources.

Backing up even a bit further, it's possible that users would eventually be working with *more than one* collection. The simplest solution is to keep them in different folders. I.e., I envision a collection-folder called `18-19cBooks`, that contains within it a) a master metadata file and b) a subfolder of prediction files.

So the first time you run the browser, you might need to start by selecting a default "collection folder," which would load automatically in the future. But you'd still have the option of changing the default collection folder, if you wanted to work with e.g., `20cSerials.` That folder would have its own master metadata file, and subfolder of prediction files.

Okay, let's suppose you've selected a default collection folder and you're looking at a window with a list of possible data sources. What do we need to know about each of them?

a) What is it? The whole collection, a single-class prediction, a multi-class prediction? Possibly this can be conveyed visually with some sort of icon.

b) If it's a prediction file, what class(es) does it define?

c) When was it created? When last modified?

d) What was it based on?

After you select a data source, you'll have the option of doing something with it. Four options were listed in the diagram. But for right now, let's focus on the first two options. I think it's unlikely that we'll finish them this summer.

1. Search. Say you select the whole collection (or a subset that represents "fiction") and you want to create a new prediction file of training data by searching for particular works by title or author. This is the problem we've already spent some time discussing; the question is, how to index into a reasonably large metadata file and return results for searches that may only partly match the title or author field.

Results might be sorted either by "closeness of match" or by "earliest first." Ultimately we probably want to give the user both options.

Browsing the results of a search, the user should be able to select volume(s) and add them to a growing prediction file, then make another search. We'll want to save the prediction file at intervals.

I don't anticipate a need to search at the page level (full-text search). If someone wants to do that, they're really going to have to go to HTRC. For now, let's assume that this function is just operating on whole volumes.

2. Sorting and/or confirming an existing prediction file. What happens after we use classification or clustering to create a prediction about the collection?

In reality, there are going to be errors in the prediction. We may want to do some manual editing before using it. The purpose of that editing could be:

a) to remove errors from the file, with the idea that we're actually going to produce a *complete* manually-edited version, or — more likely —

b) to create a manually-edited *subset* of the file that we can use as training data, to train a more accurate model, and produce a better prediction.

So we want to have two options with errors. We might simply remove them from the file, or we might leave them in the file but mark them as possessing 0% probability, so that they can be used as negative training examples.

We may not be able to manually search through a complete prediction file. If the file is large, we may only be able to create a manually-screened subset. So we'll want to be able to sort the file in order to select the examples likely to be most useful for training. The only field we're likely to sort on is probability. We should be able to sort either in ascending or in descending order of probability.

The main interaction screen here is simply a list of data objects (volumes or page ranges) allowing the user to check one of three boxes — accept and set probability to 100%, accept and set probability to 0%, or remove from prediction. Hopefully we can fit a very short version of author, date and title on this screen to characterize each volume.

But we might want to click on a data object to get more information. This could be either a) more information about the volume (like a fuller title) or b) actually inspecting pages. If we get to (b) this summer (very doubtful!) we may have to mock it up, because it'll take Boris a while to put my page-level data in an API.