Appendix A: Hybrid Curation Example

Below is an attempt to walk through a notional example of the steps necessary to apply the techniques and software described in this whitepaper.

Steps

1. Gather documents and annotations
2. Merge annotations
3. Create task items
4. Bundle items (optional)
5. Design HIT template
6. Create HIT on MTurk
7. Post the batch of HITs
8. Download results
9. Unbundle (optional)
10. Aggregate
11. Evaluate

Gather documents and annotations

We assume that you have a set of documents you wish to curate, and various annotations on those documents. There is typically cleanup and other conditioning of these datasets necessary before proceeding, but these are outside the scope of this example. We assume that each annotation has the following elements:

- Doc ID
- Start and end
- Concept ID
- Content string

Merge annotations

In the typical case explored in our research, there are multiple kinds of annotations, e.g. genes and mutations, or drugs and diseases. These are often the result of distinct taggers or extraction processes, and so the first step is to merge all the various types of annotations. There are several senses in which annotations must be merged. First, all of the annotations associated with a particular document must be grouped together, regardless of their type. And second, all of the annotations representing the same concept must be grouped together.

In our experiments we use Python script named simple-merge.py. This takes a set of documents and multiple annotation files, and produces a JSON representation of each document along with all of the annotations of that document, grouped by their type and then by their concept ID. simple-merge will optionally produce a gloss for each of the concept groups by picking the most frequently occurring content string.

Create task items

The next step is to produce the actual items that will be presented to the workers. For each document, this is the cross-product of each distinct concept annotated in the document. For example, a document may have two drugs annotated (each with multiple occurrences), and three diseases. In this case, there should be six items generated for that document. In our experiments we use a script called make-items, which takes the result of simple-merge and performs exactly the cross-product described above for each document. The result is a JSON representation of each item containing the entire text of the document, with the relevant annotations now inserted inline.

Bundle items

There is something to an art to scaling MTurk HITs to require the appropriate amount of effort and time. If an item takes only a few seconds, it may make sense to bundle a number of items together into an individual HIT. In our hybrid curation work the items often require scanning a paragraph or more of text, and so typically a one-to-one correspondence between items and HITs makes sense. Nonetheless, we sometimes use a script called bundle-items to partition the items into fixed size subsets.

Probably more relevant to hybrid curation, bundle-items can be used to mix control items into the HITs in a specific proportion. The script takes as input a set of test items, an optional set of control items, and a numeric bundle size and test-item ratio.

Design HIT template

The MTurk API can be used to upload individual HITs, but the simplest approach is to design an HTML file with placeholder variables that will be filled in automatically from a CSV file. (This is similar to the *mail merge* functionality provided by many word processing platforms.) In the case of the typical hybrid curation task, this might be a paragraph of content, with each entity in the candidate relation highlighted. HTML form elements are used for the Turkers to respond to the HIT.

Create HIT on MTurk

When using the MTurk web site to manually manage batches of HITs, you specify a few details, such as the name of the HIT type, and short descriptive text that the Turkers will see. You also indicate how much redundancy you want, how much you will pay per HIT, and optionally other requirements such as qualifiers, country limits, etc. You then can cut and paste the HTML content into the editor for the HIT design. Alternatively, the HIT template can be uploaded with the API.

Post the HITs

The final step to posting the HITs is to upload a CSV file with the data that will fill in the placeholder variables in the HTML form the previous step. Each row in the CSV file will be a separate HIT on the MTurk site, with the columns used to instantiate a different placeholder variable.

Download

When all of the HITs have been completed, you can use the MTurk website to download the results as a CSV file. This will have a line for each HIT performed by each Turker. It will have the same fields as whatever you uploaded in the previous step, prepended with the string “Input.”. The CSV file will also have a number of extra fields corresponding to the Turkers responses, as well as metadata fields such as how long it took the Turker to perform that HIT, etc. The response fields will correspond to the HTML form elements in your template.

Unbundle

If you used bundle-items, then each line in the response file will correspond to multiple items. In this case you will likely want to use unbundle-items to split these out onto separate lines. At this point we typically also convert from CSV to a JSON format.

Aggregate

If you want to find the "best" answer for each of the items, you will need to somehow combine the possibly different answers from multiple Turkers. The naive-bayes script implements a simple aggregator that computes Bayes factors for each Turker based on their control item responses, and then "votes" their responses accordingly.

Evaluate

If you are interested in how the Turkers performed, either individually or in aggregate, you can use simple-score to compare their responses to an answer key.