Rich Context Competition - Competition Design Chapter

By Andrew Gordon, Ekaterina Levitskaya, and Jonathan Morgan - New York University

# Table of Contents

[**Table of Contents**](#_ylg1bxveq1hh) **1**

[**Competition Design**](#_t4fu058w7yiq) **2**

[**Data**](#_d2p691yk4pgj) **3**

[Publications](#_szaaltlpzqlm) 3

[Publication Dataset - Phase 1](#_tta2md872rwx) 4

[Publication Dataset - Phase 2](#_9m20d3x7ftda) 5

[Converting PDF files to plain text](#_gg5g4dxnp3ey) 5

[Data Sets](#_ugwsa86kti6v) 6

[Data Set Mention Annotation Process](#_vofr8k96bcvl) 8

[Methods and Fields](#_27lk2j4bhmjt) 11

[**Developing a submission process**](#_dqttnbq4m14h) **12**

[Building and Submitting a Model](#_7rvxrdr7210a) 13

[Model API](#_98a579t3sqwq) 14

[Running a Submitted Model](#_rrju297x8yat) 15

[Notes on the Submission Process](#_kbnm9toxvops) 16

[**Evaluation**](#_7ecvqct6hgg8) **17**

[Phase 1 Evaluation](#_n7wikrprow43) 17

[Mentions, Methods and Fields](#_bas8ppnn3pbl) 17

[Data Set Citations](#_1k9s7jxkss6k) 17

[Phase 2 Evaluation](#_q2aauht98w9) 18

[Data Set Citations](#_r73urongb842) 21

[Capturing Data References](#_m501v7u0ebm8) 21

[Finding Related Mentions and Citations](#_4uyjq9vm6yiw) 22

[Scoring the Results](#_bf3uyc2nu7zb) 24

[Notes on Phase 2 Evaluation](#_mssi3f438a7s) 24

[**Conclusion**](#_g9hhit9tm6nu) **25**

# Competition Design

The rich context competition was designed as the start to a series intended to inspire data scientists to use artificial intelligence and machine learning to develop and identify text analysis and machine learning techniques to discover relationships between data sets, researchers, publications, research methods, and fields. Specifically, the goal of this first round of competition was to use any combination of machine learning and data analysis methods to identify the datasets mentioned in a corpus of social science publications and infer both scientific methods used in the analysis and the publication’s research fields.

The competition had two phases.

In the first phase, participating teams were provided with a listing of datasets and a labeled corpus of 5,000 publications with an additional dev fold of 100 publications. Each publication was labeled to indicate which of the datasets from the master list were referenced within and what specific text was used to refer to each dataset. The teams used this data to train and tune algorithms to detect mentions of data in publication text and, when a data set in our list is mentioned, tie each mention to the appropriate data set. A separate corpus of 5,000 labeled publications was held back to serve as an evaluation corpus. Each team was allowed up to 2 test runs against this evaluation corpus before final submission. The final models of each group were run against this holdout corpus and the results were used to evaluate submissions, along with a random qualitative review of the mentions, methods, and fields detected by the team’s model. Submissions were primarily scored on the accuracy of techniques, the quality of documentation and code, the efficiency of the algorithm, and the quality and novelty of the methods and research fields inferred for each of the publications.

Four finalist teams were selected to participate in the second phase, the teams from: Allen Institute for Artificial Intelligence, United States; GESIS at the University of Mannheim, Germany; Paderborn University, Germany; and KAIST in South Korea.

In the second phase, finalists were provided with a new training corpus of 5000 unlabeled publications and asked to discover which of the datasets from the first phase’s data catalog were used in each publication, as well as infer associated research methods and fields. As in the first phase, teams were scored on the accuracy of their techniques, the quality of their documentation and code, the efficiency of their algorithm, and the quality and novelty of the methods and research fields inferred for each of the publications.

At the end of each phase, competing teams packaged their models into a docker container using a model packaging framework designed and built for the competition by NYU, and the containers were installed on AWS servers and run by the competition organizers against the holdout to generate predictions that were used to evaluate the teams.

# Data

In each of the two phases, competing teams were given text and metadata for 5,000 publications and single set of metadata on 10,348 data sets of interest, shared between the two phases, for use in training and testing their models. Separate 5,000-publication samples were provided for each phase. The corpus of 10,348 data sets included data maintained by Deutsche Bundesbank and the set of public data sets hosted by the Inter-university Consortium for Political and Social Research (ICPSR). In addition, a single 100-publication development fold was provided separate from the training and testing data to serve as a test for packaging of each team’s model, and as a quick test of their model and the quality of its output. For details on the metadata provided for each type of data, see <https://github.com/Coleridge-Initiative/rich-context-competition/wiki/Dataset-Description>.

In each phase, an additional separate set of 5,000 publications were held back and used to evaluate the models. After the 1st phase, the phase 1 holdout was also provided to phase 2 competitors to serve as additional training and testing data.

In phase 1, both the train-test publications and the holdout publications were broken into 2,500 publications each that used one or more of the data sets of interest for analysis, as compiled by ICPSR and Bundesbank staff, and 2,500 publications that had not been annotated and had been filtered to not contain data. The data set citations were captured in a separate data set citations JSON file. The citations for the phase 1 train-test publications were provided to competition teams to use as training data, while the citations in the phase 1 holdout were used to test the quality of each team’s model in phase 1, and given to teams as additional training data in phase 2.

In phase 2, teams were provided with the phase 1 holdout for additional annotated training data, and then provided with an additional un-annotated set of 5,000 publications to assess their model’s behavior on un-curated data. The phase 2 holdout of 5,000 publications was also unannotated, and evaluation of data set detection was based on hand-coded data set reference data revised to make the data more representative of what the models were asked to detect.

## Publications

All publication text provided to teams was either open access, and so freely available, or licensed from the publisher for use in the contest by contest participants. In each phase of the competition, a set of publications was provided to the participants and a separate set of publications was held out and kept in reserve so it could be used to evaluate the teams’ models. For each publication, participants were provided with PDF and plain text versions of each publication along with basic metadata (pub\_date; unique\_identifier - DOI or equivalent; text\_file\_name; pdf\_file\_name; and publication\_id - the unique identifier from our internal system used to manage the data, metadata, and underlying relationships between publications and data sets for the competition).

One challenge of note: Copyright and licensing around research publications limited what publications could be accessed, licensed, and distributed for the competition, and so our universe of publications was limited to publications that were either open access, or published by Sage Publications.

### Publication Dataset - Phase 1

* 2500 labeled training publications
* 2500 unlabeled/no-dataset training publications
* 100 publication development fold
* 2500 labeled holdout publications
* 2500 unlabeled/no-dataset holdout publications

In phase 1, 5,000 publications were provided to participants as a train-test data set, 5,000 publications were held back for evaluation, and 100 publications were provided as a separate development fold, for basic model testing and evaluation. The train-test and evaluation holdout each contained 2,500 publications that cited at least one data set, and 2,500 publications that had not been cited by ICPSR as using their data, and had been filtered to not have obvious markers of using data.

The annotated portion of these two sets of publications were drawn from a set of publications provided by Bundesbank that referenced their data and the publications captured in the ICPSR catalog annotated as having used a particular data set for analysis. These publications were collected in a database application designed to facilitate a mix of human and automated content analysis of publications. They were then filtered into two sets: those that were open access, and so could be shared publicly, and those that were not open access, but that were available from our publisher partner (Sage Publications, or “Sage”). Of the 5,100 total publications with annotated data citations provided to phase 1 participants, the 2,550 publications in the train-test corpus (2,500) and development fold (50) were randomly selected from the open access set, so they could be distributed freely to all participants. The 2,500 in the holdout were randomly selected from the remainder of the open access set plus those available from Sage. The un-annotated publications used in phase 1 were all published by Sage - the 2,550 non-annotated publications in the train-test corpus (2,500) and development fold (50) were open access publications from Sage journals. The 2,500 un-annotated publications used in the holdout evaluation corpus were sampled from across Sage Publications’ journal holdings including non-open access journals.

### Publication Dataset - Phase 2

* The main publication corpus for phase 2 of the competition was 10,000 unlabeled publications evenly distributed between 6 key topic areas (Education, Health care, Agriculture, Finance, Criminal justice, and Welfare), nicknamed the “wild corpus”.
* 5,000 of these 10,000 were given to teams to work with in phase 2 (randomly selected from within each of the 6 key topic areas to maintain even distribution across topic areas).
* The other 5,000 publications were held out to serve as an evaluation corpus.
* In addition, teams were given the same 100 publication development fold as in phase 1.
* Teams were given the 5,000 publication evaluation corpus from phase 1 to serve as further train-test data.

In phase 2, we worked with Sage to find publications in six key topic areas of interest for partners and future projects (Education, Health care, Agriculture, Finance, Criminal justice, and Welfare). For 28,769 matches, Sage provided PDFs for each and we parsed the text (see details below), removing any that did not parse, or that resulted in file sizes smaller than 20KB, reducing the size of the sample to 25,888. We looked at publication year and type to see if we needed to filter out older publications or non-academic publications, but there were few enough of each class (644 pre-2000 publications and 3,115 non-research articles) that we decided we’d keep all in to preserve as much potential for heterogeneity as possible. From these 25,888 publications, we then randomly selected a total of 10,000 with the goal to keep the distribution across the 6 topic areas equal (so 1666 randomly selected in 2 topic areas, 1667 randomly selected in the other 4). Then, we split the phase 2 corpus to give half to participants and keep half back for evaluation, maintaining equal distribution between the topic areas within each set of 5,000 publications.

### Converting PDF files to plain text

The plain text provided for each publication was derived from that publication’s PDF file by the competition organizers. It was not intended to be a gold standard, but to serve as an option in case a team preferred not to allocate resources to PDF parsing.

The articles were converted from PDF to text using the open source “pdftotext” application, an Xpdf text extraction system. The basic conversion used the “raw” mode of “pdftotext”:

pdftotext -raw <path\_to\_pdf.pdf> <path\_to\_txt.txt>

There are many approaches and tools available for this task. The rationale behind this simplified process for converting pdfs to texts:

1. To render the most usable txt files from available pdfs without over engineering for any specific types of pdf files (e.g., single column vs. multi-column).
2. To have a process that is easily reproducible across different machines for free. That is, not all PDFs convert the same way. Some are more error prone than others. More advanced OCR techniques might have been able to compensate where Xpdf might have fallen short, but relying on more sophisticated and perhaps costly text conversion processes would have made the conversion pipeline more expensive to reproduce and less portable across different applications.

Because of the basic approach, there were some limitations to note:

* Many artifacts from PDF formatting were left behind in the text.
* We had to tweak our processing to get multi-column layouts to output text in order in a linear, single-column text output, and the method we ended up using to achieve this precluded more nuanced processing of other elements of the PDFs.
* Example: tables and charts were not converted in any way to text.

Competition participants were encouraged to try their own conversion process if this text did not meet their needs. If participant teams chose to use another means for converting PDF files to plain text, we asked that they supply us with documentation for installing and running their conversion process so we could start to build up a set of PDF processing strategies that could be reused in the future.

## Data Sets

Competitors were provided with two sets of data related to detecting data sets: 1) a catalog of all of the data sets of interest that models were tasked with finding in publications, including basic metadata for all and a list of verbatim mention text snippets for those that were cited in the train-test data; and 2) a subset of these data sets that were actually specifically annotated as having been used for analysis in a given publication.

The data set catalog, provided to participants in the JSON file data\_sets.json, contained metadata for all public datasets in the ICPSR data repository and a subset of public data sets available from Deutsche Bundesbank. It includes all data sets sited in the train-test and evaluation corpora, plus many others not cited in either. The data was provided in JSON format for ease of use, a JSON list of JSON objects, each of which contains:

* subjects - list of terms associated with the dataset, based on the [ICPSR subject thesaurus.](https://www.icpsr.umich.edu/icpsrweb/ICPSR/thesaurus/subject)
* additional\_keywords - System keyword for where dataset originated.
* citation - Preferred dataset citation.
* data\_set\_id - Integer ID for dataset from our internal data store of publications, data sets, and relations. This is the identifier used in the data\_set\_citations.json file to identify relationships between datasets and publications.
* title - Canonical title for dataset.
* name - Canonical title for dataset.
* description - Dataset description, if available.
* unique\_identifier - Original unique identifier for dataset, normally a DOI if available.
* methodology - Methodology for dataset, if available.
* date - Date when dataset was published, if available.
* coverages - Geographic coverages, if available.
* family\_identifier - Internal system ID, roughly captures datasets that have multiple years but are the same dataset. Inconsistently applied, should not be used in analysis.
* mention\_list - Array of strings for annotated mentions as identified by human reviewers. Not an exhaustive list of mentions for any given dataset, and only populated for those data sets cited in the phase 1 train-test corpus.

The mention list is the superset of all unique mention strings associated with each data set across all of that data set’s citations where mention data was created. Mention data was only created for data sets cited in the phase 1 train-test corpus.

ICPSR captured when a given data set was used in analysis within a particular publication, but it did not capture particulars on how that determination was made. To provide better data for participants, we implemented a human content analysis protocol to capture mention text for each data set-publication pair included in our train-test corpus (see [Data Set Mention Annotation Process](#_vofr8k96bcvl) below). Since we manually created this data, given limited time and resources, we initially only did this work for data sets that the teams would be using for training and testing in phase 1. In future work, we intend to provide this kind of information for all data sets of interest, and to refine the protocol to capture the exact position in the text of each mention along with the verbatim text.

Citations of data sets by publications within our phase 1 corpora were captured in separate data\_set\_citations.json files for each of the train-test and evaluation corpora. Each of these JSON files contains a JSON list of JSON objects, each of which specifics a single relationship between a data set and a publication. This JSON format is also used by models to output detected citations. Each citation contains:

* citation\_id - A unique ID for the relationship between one dataset and one publication
* publication\_id - Unique ID for a publication which is the same ID for the publication in publications.json
* data\_set\_id - Unique ID for a dataset which is the same ID for the dataset in the data\_sets.json file.
* mention\_list - Optional array of strings for alternative references for the dataset in the specific publication (only present in citations included in train-test corpus, and even then, could still be empty).
* score - Confidence score for the dataset being found in the related publication. In ICPSR-specified citations, the score will be 1.0. In model-created files, will depend on the model.

Even citations from the phase 1 train-test corpus could have an empty mentions list. A given publication could, for example, have been tagged with a dataset by the curator (either at Bundesbank or ICPSR) based on knowledge of the publication and dataset, but a human coder without this knowledge was not subsequently able to find specific mentions within the publication, or the human coder could simply have missed the references. An empty mentions list is not a guarantee that the data set in question was not mentioned.

The list of data sets cited in a particular publication is also not exhaustive. There is the possibility that other data sets from our catalog of data sets of interest were used in analysis within a paper but not captured. The ICPSR data did not include mentions where data was not used in analysis, even of other ICPSR data sets. And named data sets not within our catalog of data sets of interest could also have been used in analysis within a given publication.

### Data Set Mention Annotation Process

One long-term goal of our efforts in data set detection is to build generalized models that are not overly dependent on use of formal titles of data sets. We aim for models that know of and use the language of discussing and using data to recognize where data is discussed in a particular article and then identify which data sets. The ICPSR data contains many explicit ties between publications and data sets that would have been hard to come by otherwise, but the lack of any indication of which parts of the publication indicated the citation relationship made it difficult to identify the linguistic context within the publication that captured the relationship.

To make it easier for participants in the competition to efficiently and systematically engage with the language used to discuss data, we developed a content analysis protocol and accompanying web-based coding application so human coders could examine all of the data set citations in our train-test corpus and capture mention text for each. This required human workers to examine each data set citation in the context of its publication (there were X citations in 2500 training publications) to identify and mark locations in the text where each data set was referenced.

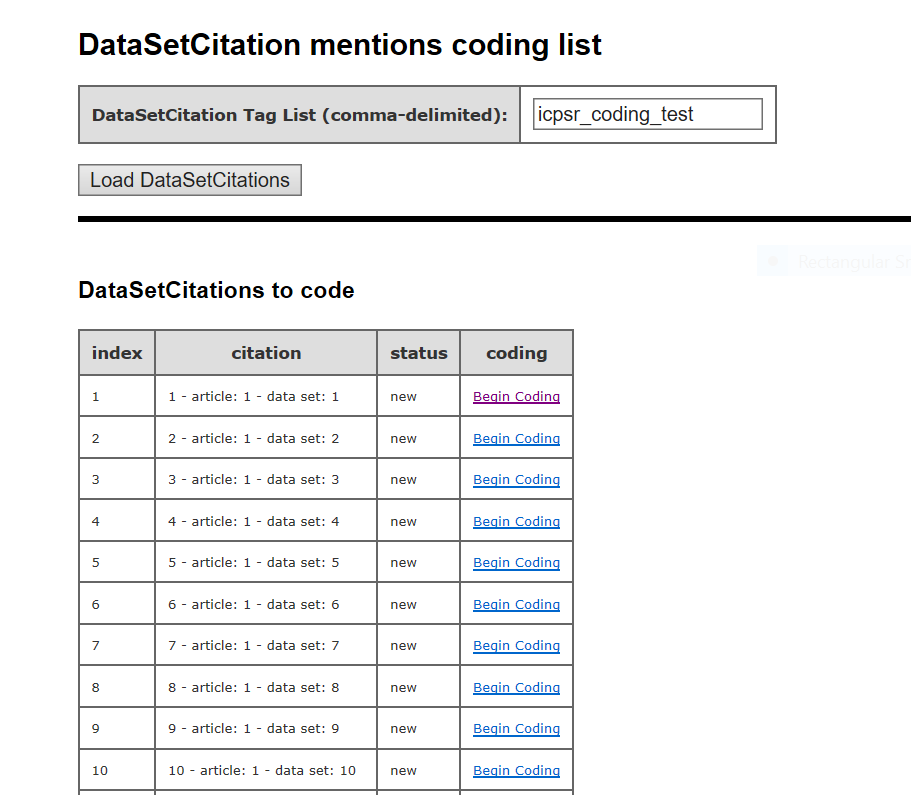
Because of the manual effort required, we only did this for the 2,500 train-test publications that referenced data provided to the teams. We did not manually annotate mention text in the 2,500 publications in the phase 1 holdout, and this made that data a little less useful for teams when it was given to them in phase 2.

Our team of coders was spread across the United States, and so we used a web-based application with a central database store to allow our distributed team of coders to work in parallel. The basic unit of work was a publication-data set pair (so a given publication would be examined as many times as it had different data sets cited within it).

The ICPSR data set repository is very fine-grained in definition of a data set, so each year of an ongoing survey, for example, might have its own data set. To save time, we eventually created the concept of a data set family for these types of data sets and assigned coding for any one instance in a family to all other instances from that family within a given publication. So, for example, multiple years of the same survey or longitudinal data collection were related to each other in a family, and then coding for one year within a paper was used for all other years cited in that paper.

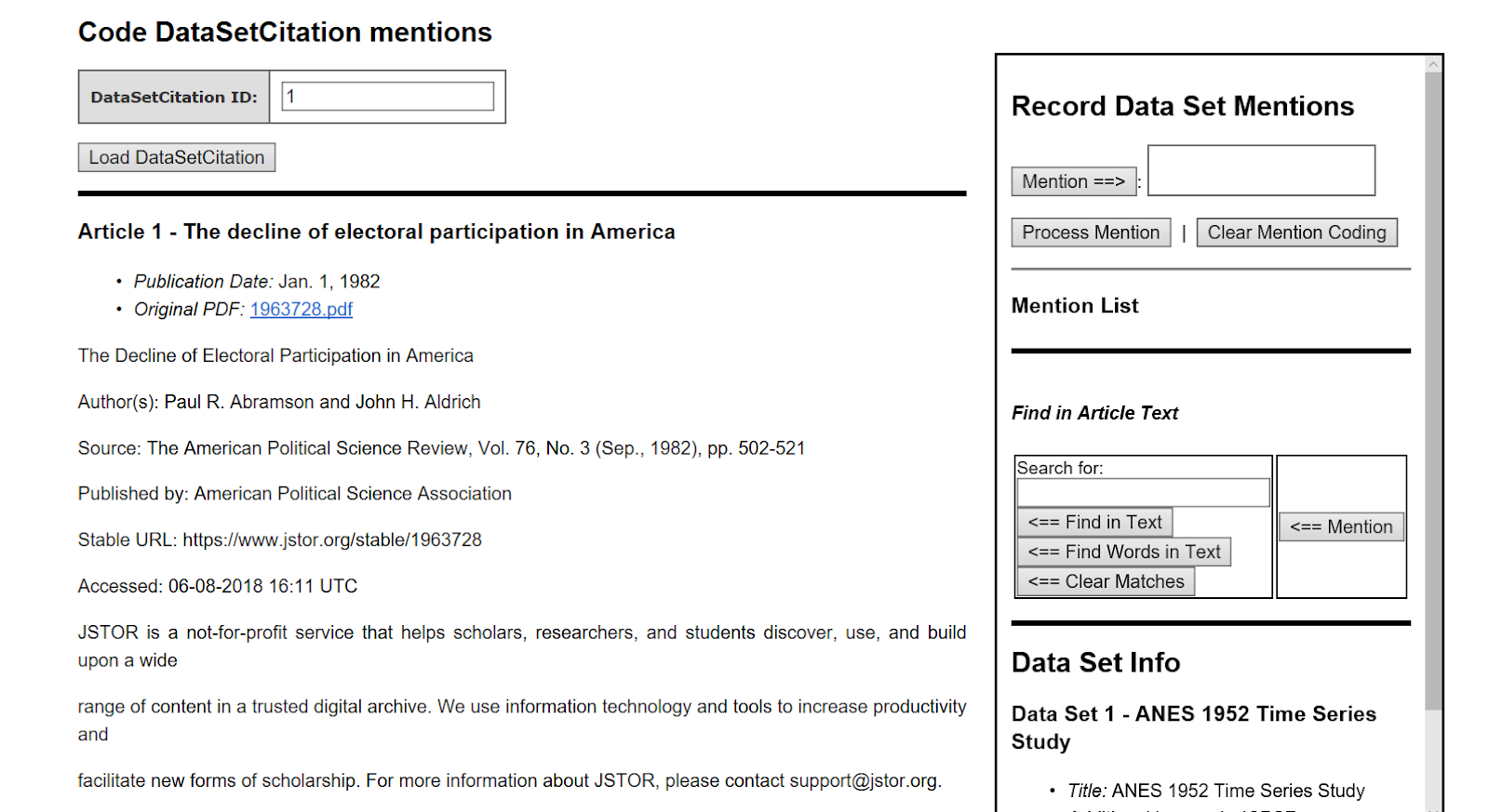
The general process:

* each user was assigned a list of citations to code.
* Once the user logged in to the coding tool, they were presented with a list of the coding tasks assigned to them that included a status of each, so they could track which they had already completed, and a link for each to the coding page:



*Figure 1. The start page of the coding tool with a list of publications with dataset citations to code. The table shows the ID of the citation and status. If the publication hasn’t been coded, the status will say “new”. Choosing “Begin Coding” option takes a user to the coding page (Figure 2).*

* Once the user loads a particular citation for coding, they are presented with the following coding page, and are asked to follow the coding instructions in the codebook/documentation for the annotation tool (<https://docs.google.com/document/d/1xuZL_-z1re6TO3Sv8_9tdFk7z6ovyqTwDVgc1bYO3Ag/edit>):



*Figure 2. The interface of a given publication and a mention capturing process in the coding tool. The**left pane contains a full text of an article to code. The right pane contains the coding interface at the top. The “Data Set Info” section contains basic metadata on the data set (title, date of collection, formal identifiers), as well as a list of synonyms gathered so far from publications where the data set is cited.*

Coders selected mention text on the left, clicked the “Mention ⇒” button to pull it into coding are, then clicked “Process Mention” to add it to the list of mentions of the data set within the publication. On loading, the coding page automatically highlights the words in the data set’s title and related terms within the publication text. It also highlights other instances of a given mention when one is selected, and has an area for doing free-form searches in the text if desired.

Coders were instructed to find terms that relate to mentions of the dataset and avoid general synonyms of those terms (for example, tagging “ANS survey” instead of only “survey”). If the phrase provides additional information about collection of the dataset, the mention is tagged twice. For example, in the case of “ANS survey collected/conducted by X”, “ANS survey” is captured first, and then “ANS survey collected/conducted by X”. At the same time, we tried to avoid including too much descriptive information of the dataset - the task is just to code the specific mentions of a particular dataset, including alternate names (e.g. abbreviations, etc.), rather than trying to capture full text in which the data set is discussed.

For more details, including an FAQ that provides guidance on specific issues that arose during coding (like how to deal with data sets that span multiple years), see the content analysis protocol: <https://docs.google.com/document/d/1xuZL_-z1re6TO3Sv8_9tdFk7z6ovyqTwDVgc1bYO3Ag/edit>

In total, a team of 5 coders, with a background in text analytics for policy research and computational linguistics, completed the task (Emily Wiegand, Neil Miller and Jenna Chapman from Chapin Hall at the University of Chicago, Mengxuan Zhao, Marcos Ynoa and Ekaterina Levitskaya from the CUNY Graduate Center, Computational Linguistics program). The results were then used to re-render data\_sets.json and the data\_set\_citations.json file for the phase 1 train-test data to include mentions.

This combined protocol and tool were developed in-house. Considerations behind building in-house:

* From previous work, we had an open-source tool that did what we would need with minor tweaks, so were able to leverage substantial existing work, though we did have to pay for the work to customize it as well as the AWS t2.large instance on which we hosted it.
* This tool includes templates for human-coding application pages like the one we used, but it is also designed to be used to build up data about publications from multiple sources and this data is straightforward to query and interact with. This allowed us to use the underlying database and application code as the competition dataset database, not just a place to handle mention coding.
* We looked at off-the-shelf text annotators and Qualitative Analysis tool such as lighttag.io, tag.works, NVivo, Atlas.ti, MAXQDA. Unfortunately, given a tight timeline and relatively complex requirements, we didn’t have the time to come up to speed with any of these tools. In addition, we needed the tool to be usable by a distributed team, and that precluded some tools above that did not support distributed workflows.
* For future coding work, we would love to be able to outsource coding tool development, and so are looking at distributed coding applications like lighttag.io and tag.works.

## Methods and Fields

For the task of detecting methods and fields for a given publication, our goals were broader than simply providing a vocabulary for each and asking the teams to classify publications against them. We want to encourage development of models that not only can figure out when a given publication is a part of an existing field or uses an existing method, but that also understand enough about fields and methods such that they can be used to detect new fields and methods as they emerge, and can then be used to look back through time for traces of these new fields and methods to track their growth and evolution.

To support this goal, we did not give any formal set of either methods or fields that participants needed to train models to classify from. Instead, we provided examples of taxonomies of methods and fields that Sage Publications uses to classify their publications, and we directed participants to use them as an example, but to try to make models that would be more creative and potentially able to find new, emerging, or novel fields rather than just fit a publication to a term from a predefined taxonomy.

In practice, this decision to forego any kind of fitting to an existing taxonomy showed the complexity of the problem of understanding fields and methods well enough to detect them based on linguistic context, rather than classifying to an existing vocabulary. Some teams limited themselves to the vocabularies we defined, and the results were uninspiring. Some teams tried to detect based on text, but ended up with a lot of noise and few relevant terms.

In addition, we also learned that there is complexity in “methods” that lumping all methods together did not account for: methods could mean many things, and we started to find sub-categories that we wish we had broken this into: statistical methods, analysis methods, data collection and creation methods, etc.

For future work, for each of these types of information, we intend to first work to decide what exactly we mean by “fields” and “methods”, then find or develop one or more taxonomies to precisely capture what we mean. Once we have these taxonomies, we’ll focus separately on building models to classify publications to them, and making models to extend and update them.

# Developing a submission process

The primary goal of the submission process developed for our competition was to balance the effort needed for a particular group of participants to package their model for submission with the effort needed from the competition organizers to configure, run, and troubleshoot submissions once they were received.

More specifically, we had the following requirements:

* Create submission infrastructure to make it as straightforward and easy as possible for a team to package their model for submission, including minimizing the understanding needed to use technologies chosen for packaging and deployment and having a built-in way to automatically run the model over the dev fold to validate processing of standard input formats and creation of required output formats.
* Minimize the installation and configuration work needed on part of competition organizers to replicate computing environments as part of model submission process.
* Maximize our ability to see and be able to test how each submission environment is set up, and so avoid accepting a blackbox that could contain anything (including malicious code or sneaky/clever tricks).

## Building and Submitting a Model

Our approach for participants building and submitting a model combines Box.com, docker, a git repo for code to implement and support infrastructure, and shell scripts. The central workspace for competition participants was a Box folder that contained example docker files, a copy of the dev fold, and shell scripts that implemented the basic steps of packaging, building, running, and testing a model. The git repository (<https://github.com/Coleridge-Initiative/rich-context-competition>) was integral to our framework, but was not used directly by participants. Its code repository was solely used as a home for the code, scripts, and files that made up our submission framework. We did, however, host documentation for participants in the repository’s main README and its wiki (<https://github.com/Coleridge-Initiative/rich-context-competition/wiki>).

To get started, participants downloaded a compressed archive of the Box folder and extracted it onto a system with a bash shell. Windows systems were supported, but we recommended that participants with Windows machines work inside a linux virtual machine.

This work folder contained:

* the script “rcc.sh” and its accompanying configuration “config.sh”, that implements all of the basic actions needed to manage docker for a model.
* A set of scaffold files and folders that demonstrate how to hook a model into a docker container, including a Dockerfile with examples of installing OS packages and python packges in a docker container and an example “project” folder with a “code.sh” shell script that is called by default when the docker container is run, pre-configured to call a provided example python file named “project.py”.
* A copy of the git repo, for use by the scripts.
* A copy of the dev fold, in the standard data folder structure.

The set of scaffold files provided out of the box could be used along with “rcc.sh” to create a simple docker container to test one’s local install of docker (including reading from and writing to a data folder configure in “config.sh”, running a script in the work folder, and creating output).

Participants were then instructed to work within the “project” folder in their work folder, get their code working first on their local machine, then set up a docker container using the provided example files and get the model running there, to isolate problems with docker from problems with their model.

When participants were ready to submit, they were asked to compress their work folder and upload it to the root of their group’s project folder and send an email to the organizers.

Participants were allowed 2 test submissions before the final submission, and most groups took us up on those test submissions in phases 1 and 2. All groups were able to work within the “code.sh” and “project.py” files in “project” to get their model to run, so no further customizations were needed.

## Model API

Our submission framework used a file-system based API for giving the model input and accepting output. We interaction through the file system to keep the configuration and implementation simple.

Each time the docker container for a model is run, it is configured to work in a particular data folder.

This data folder has a standard directory structure:

data  
|\_input  
| |\_files  
| |\_text  
| |\_pdf  
|\_output

All input information is stored in the “data/input” folder. All output is expected to be stored in the “data/output” folder.

The input folder will contain a "publications.json" file, with the same contents as described above in the “Data → Publications” section of this chapter, that lists the articles to be processed in the current run of the model. Publication plain text is stored in “data/input/files/text”, one text file to a publication, with a given publication's text named "<publication\_id>.txt". The original PDF files are stored in “data/input/files/pdf”, one PDF file to a publication, with a given publication's text named "<publication\_id>.pdf".

The output folder starts out empty, and is where the model is expected to place 4 output files after each run of the model:

* **data\_set\_citations.json** - A JSON file that contains publication-dataset pairs for each detected mention of any of the data sets provided in the contest data\_sets.json file. The JSON file should contain a JSON list of objects, where each object represents a single publication-dataset pair and includes four properties:
  + publication\_id - The integer publication\_id of the publication from publications.json.
  + data\_set\_id - The integer data\_set\_id that identifies the cited dataset.
  + score - A score on a scale of 0 to 1 representing the level of confidence that the dataset is referenced in the publication.
  + mention\_list - A list of the text of explicit mentions of the data set in the publication.
* **data\_set\_mentions.json** - A JSON file that should contain a list of JSON objects, where each object contains a single publication-mention pair for every data set mention detected within each publication, regardless of whether a gvien data set is one of the data sets provided in the contest data set file. Each mention JSON object will includes three properties:
  + publication\_id - The integer publication\_id of the publication from publications.json.
  + mention - The specific data set mention text found in the publication. Each mention gets its own JSON object in this list.
  + score - A score on a scale of 0 to 1 representing the level of confidence that the mention text references data.
* **methods.json** - A JSON file that should contain a list of JSON objects, where each object captures publication-method pairs via three properties:
  + publication\_id - The integer publication\_id of the publication from publications.json.
  + method - The inferred method used by the research in the publication.
  + score - A score on a scale of 0 to 1 representing the level of confidence that the method is used in the publication.
* **research\_fields.json** - A JSON file that should contain a list of JSON objects, where each object captures publication-research field pairs via three properties.
  + publication\_id - The integer publication\_id of the publication from publications.json.
  + research\_field - The inferred research field of the research in the publication.
  + score - A score on a scale of 0 to 1 representing the level of confidence that the publication is in the stated research field.

## Running a Submitted Model

Once a model was submitted, the competition organizers followed a standard script for running the model and processing its output for analysis:

* For each submission, an AWS instance was spun up from a standard image pre-configured to run models built using our submission framework.
* The evaluator connected to the instance and started a screen session, so work would not be disrupted if connection to server was lost.
* The model was downloaded to the server and extracted.
* The submission container was built on the server using the provided Dockerfile and “rcc.sh”, and then the container was run over the dev fold to test basic functionality of the container and the model, and to give an estimate of time needed to complete.
* Once the dev fold was successfully processed, “config.sh” was reconfigured to point at the evaluation corpus, and the model was run over the evaluation corpus.
* Once the model completed, standard evaluation Jupyter notebooks in the git submission framework repository were configured to the current projects output and run to generate materials for judges to evaluate the submission.
* Output and results were copied to a central storage area, and the instance used to run the model was terminated.

Throughout this process, the evaluator communicated any problems with the participant team and worked with the team to address problems and turn around a new version of the model as quickly as possible. If a team’s model performed poorly on the standard size machine, we also would sometimes try different sizes of server to give them an idea of whether their problem was related to needing more compute power, or was a limitation of their approach independent of available resources.

## Notes on the Submission Process

We chose Box.com because we have unlimited space there through NYU, and so we were able to accommodate not only whatever data participants needed to provide to make their models work, but also all of the data we provided to participants for training and testing. To minimize confusion, we pre-configured and shared each team’s Box folder with them, so they did not have to do any setup.

To setup the infrastructure in each folder, we created a git repository (https://github.com/Coleridge-Initiative/rich-context-competition) that contained all of the files, shell scripts, and templates needed to: 1) configure a new instance of a team folder, for use by competition staff setting up team folders; 2) develop, package and test deployment of a model (participants); and 3) support building, running, and evaluating the models once they were submitted.

We considered using github to store participant submissions, but ended up needing to use something like Box.com that did not have limitations on storage. We also were OK with Box rather than github to minimize the potential support cost of participants with little experience with git and github.

We also seriously considered using an external service for hosting the competition like Codalab or Kaggle, but an initial assessment of each suggested that they would not meet our needs without substantial changes to the design of our competition:

* Codalab looked promising, but its documentation was sparse and our time frame was short enough that we weren’t comfortable we could get up to speed with it quickly enough to make a reliable, easy-to-use competition with it.
* Kaggle seemed designed for more basic competition designs (our evaluation steps were fuzzy, so couldn’t just take their outputs and make scores - this is not entirely a knock on them - it would be great to get our tasks to the point where they fit in this framework, we just don’t have the data yet), and there were also licensing complications we weren’t comfortable sorting out. We also needed control over manual evaluation and were concerned there that their submission and evaluation system wouldn’t support the bespoke nature of our submissions.
* For both, we also simply weren’t comfortable that we’d be able to get up to speed on the platform in time to make the experience of participating in the competition as pleasant and painless as possible.

We also wanted to have the flexibility to run many models in parallel and give models substantial resources if needed, to see how they performed with different magnitudes of computing resources and to allow us to try to throw raw compute power at a model if it was running too slowly, to get it to complete so we could give as good of feedback as possible. We not only wanted groups to be able to do preliminary submissions, but we wanted to make sure we could give as much feedback as possible. This led us toward a container-based approach where we did what we could to abstract and simplify the running of models, and allowed for flexibility and configurability in the instances that we spun up to run the models.

# Evaluation

In both phases of the competition, we evaluated raw mentions, research fields, and research methods separate from citation of named data sets.

## Phase 1 Evaluation

### Mentions, Methods and Fields

In phase 1, expert social science judges evaluated mentions, methods, and fields in two ways: 1) we randomly selected 10 publications to manually examine each team’s output against, and made notes of good and bad for each team, then ranked the teams within each publication; and 2) we generated distributions of all values found across all publications within each type of value, counted the occurrences of each, compared the distributions across teams, and ranked the teams based on how their distributions compared. To create overall rankings, the judges met, compared notes and individual rankings, and then agreed on an overall ranking of the teams.

### Data Set Citations

To evaluate data set citations in phase 1, we used the ICPSR citation data as our evaluation baseline for creating a confusion matrix based on how each team’s citation findings compared to ICPSR’s baseline, and we calculated precision, recall, and F1 scores from the confusion matrix to compare across teams. To create the confusion matrix for each team, we started with a list of all of the data set-publication pairs found either in ICPSR’s baseline or the team’s output. We created found-or-not (1 or 0) vectors for every publication-data set pair for the baseline, and for the team. Then, for each data set-publication pair, we compared the values between the baseline vector and the team vector to decide how to update the confusion matrix for that pair: if a team agreed with ICPSR on presence of a data set, that was counted as a true positive (TP). If the team found a data set that ICPSR did not, that was counted as a false positive (FP). If a team missed a data set ICPSR indicated was present, it was counted as a false negative (FN). We did not develop a way to capture true negatives since the metrics we used to evaluate did not require it. In addition, as part of the processing to create the overall confusion matrix, we also created per-publication confusion matrices for each publication, so we could track average false positives and false negatives per publication, and highlight publications that were higher than the average, for more detailed evaluation.

This approach had some significant limitations. The ICPSR data set metadata and citation data are a perfect example of quality metadata that eventually should be enough to train a model that finds named data sets. Some traits of this data made it a less than optimal fit for evaluation of models that were trained to simply detect data sets cited in a publication, regardless of whether they were used in analysis or just discussed. The ICPSR citation data for a given publication only included ICPSR data sets that ICPSR staff had annotated as being used in analysis within the publication (so some ICPSR data sets could be missed if they weren’t the focus of the annotation). It did not include mention text where we did not create it, and so for the majority of data sets, the only text available for characterizing a data set was the title and a paragraph of description, no examples of how the data would be discussed within a publication. It did not include non-ICPSR data sets at all, and it did not include any data sets mentioned but not used in analysis, ICPSR or not. It also broke longitudinal data sets up into many separate data sets by year and sometimes even topic area within year, and this level of detail was often beyond the understanding of participants’ models.

We also decided to defer figuring out “mentioned” vs. “used in analysis” in our initial competition, to make the initial task more manageable. This decision, combined with the traits of the ICPSR data, caused substantial noise in the phase 1 precision/recall/F1 scores. For example, even models that figured out that a longitudinal data set was present sometimes got many false positives and false negatives because they got the years wrong, and models that correctly found ICPSR data sets used in discussion had those counted as false positives because ICPSR had only captured data sets used in analysis.

## Phase 2 Evaluation

In evaluating phase 2, we kept the division between mentions, fields, and methods and citations, but we refined our evaluation methods in based on what we’d learned in the first phase.

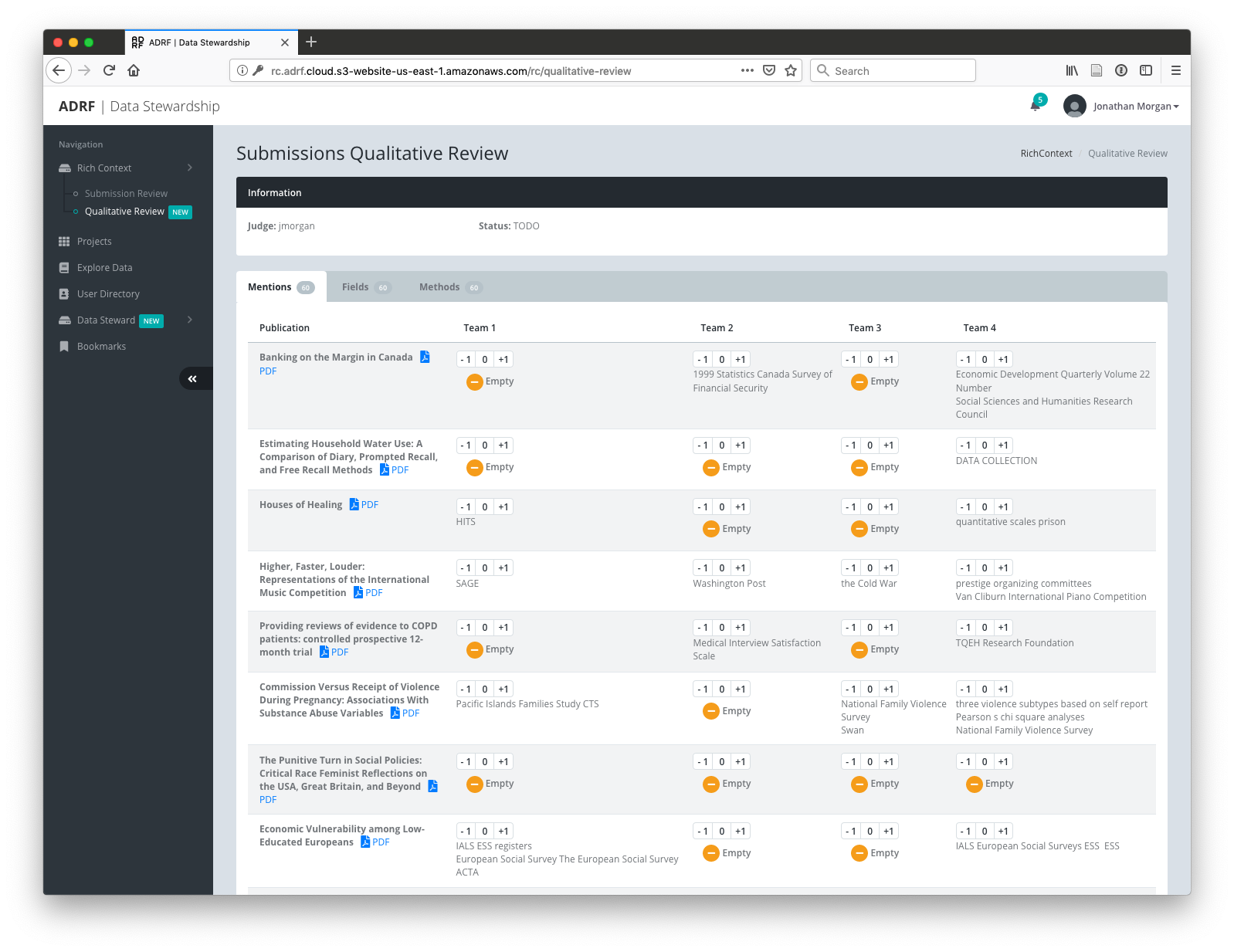
Mentions, Methods and Fields

For mentions, methods, and fields in phase 2, we kept the basic strategy of: 1) comparing the values created by each team’s model in the context of a set of selected publications and 2) reviewing the overall distributions of values for each team.

We expanded the number of publications across which we compared values to make the sample reviewed more representative, though, and created a web-based tool to help judges deal with the added work from more publications to review. We also selected publications differently for data mentions from fields and methods, choosing publications with different levels of agreement between the teams on whether data was present or not, to start to evaluate the different model’s ability to detect data at all, in addition to comparing the results when they thought a publication contained data.

For fields and methods (and data set citations), we selected 20 publications for each of our 6 topic areas of interest (Education, Health care, Agriculture, Finance, Criminal justice, and Welfare) with a few extras (2 extra in finance and 1 extra in criminal justice), for a total of 123 publications to compare values across. Within the 20 publications per topic area, we worked through a random selection of articles picking publications to add to our sample to fill out a rough ratio within each topic area of 5:4:1 between publications with titled data sets (5); data described, but not titled (4); and no data (1).

To make it easier for the judges to work through this increased number of publications, we also created a tool that collected the output for each team side-by-side per publication along with a link to each publication’s PDF, and had a place for the judge to score each team’s output for a given publication from among “-1”, “0”, and “1”. Once judges scored all output, we then created rankings based on the sum of each team’s scores.



*Figure 3: The interface given to judges to evaluate data set mentions, research fields, and research methods.*

For manual evaluation of data set mentions, we used the same tool described above, but we chose a different sample of 60 publications based on agreement between the output of the different participant team models as to whether publications had data mentions. To generate this sample, we loaded all of the output from each team’s model into our work database and then made a list of all of the publications in our phase 2 holdout and the count of teams that had data set mentions for each publication. We then sampled from within each of the distinct count values to get 60 publications:

* 10 publications where all teams agreed there was no data.
* 10 publications where all teams agreed there was data.
* And then 40 publications where the teams disagreed on whether there was data.

For the 40 publications with disagreement, we selected publications with 1 team, 2 teams, and 3 teams agreeing data was present proportional to the distribution of each level of agreement in the broader sample:

* 17 from 1 (1439/5000 = 0.2878; 0.2878 \* 60 = 17.268)
* 20 from 2 (1741/5000 = 0.3482; 0.3482 \* 60 = 20.892)
* 13 from 3 (1080/5000 = 0.216; 0.216 \* 60 = 12.96)

We then asked a separate pair of qualitative judges to use the tool to compare and evaluate the data set mentions generated by the teams across these publications.

### Data Set Citations

Our analysis of data set citations in phase 2 required a more substantial rethinking since we did not have any starting point for presence or absence of data like the ICPSR corpus.

After much reflection, for phase 2, we decided to try to implement a confusion matrix that could be used to generate precison, recall, and F1 scores more closely aligned with the task we’d assigned the teams to implement - finding mentions of data and data sets within publications.

To implement this, we started with the sample of 123 publications used for evaluating mentions and fields above and:

* Captured all “data references” within each of those publications using a new human coding protocol. This included external titled data sets either discussed or used in analysis, external data without a title that was discussed or used in analysis, and data created by the researcher for a given study.
* For each data reference, we compared all mentions and citations created by each team for the publication to the information on the data reference within that publication and marked any that were “related” to the data reference.
* Finally, we used the list of references as a baseline and built a confusion matrix based on whether each team had found mentions or citations “related” to each of the data references, along with a “false positive” record where the baseline was always 0 and the team was assigned a 1 if they had one or more mentions or citations that were not “related” to any data reference.

#### Capturing Data References

To capture data references in our sample of publications, we created a basic protocol for an initial round of data creation (<https://docs.google.com/document/d/1aFPEtT4hd93kcsOEzocyB6-a4Hu8WcemKTld-98Q25k/edit#heading=h.f3u3kdbg87s4>), then evaluated the results throughout the rest of the process. We used a single data reference coder to encourage consistency in output. Our data reference coder worked within a spreadsheet to, for each publication in our sample:

* Flag all paragraphs where data was mentioned.
* Cluster mentions together that refer to a single dataset.
* Give each cluster of mentions a row in the spreadsheet. These are our “data references”.
* Then, for each data reference:
  + Collect all mentions that refer to the reference.
  + decide if the data set is simply cited (“cited”), or if it is one used in analysis (“analysis”) in the publication
  + Capture words or phrases that are key to identification as “key terms”.
  + Also capture any broader contextual text in “Context”, so it could be used to better understand the nature of the “data reference”.
  + If data set title is present, capture it.
  + Try looking up the data set in the database, and if it is there, store its data set ID.

We tried to capture detailed context on each reference for a couple reasons: 1) To make it easier for reviewers of this data to evaluate the quality of each data reference; 2) To give more context for judges deciding if mentions and citations for a given team were “related” to a given data reference.

#### Finding Related Mentions and Citations

After the data references were captured, a team of coders then looked at each data reference team-by-team to see if data set citations and mentions by the team were “related” to the data reference.

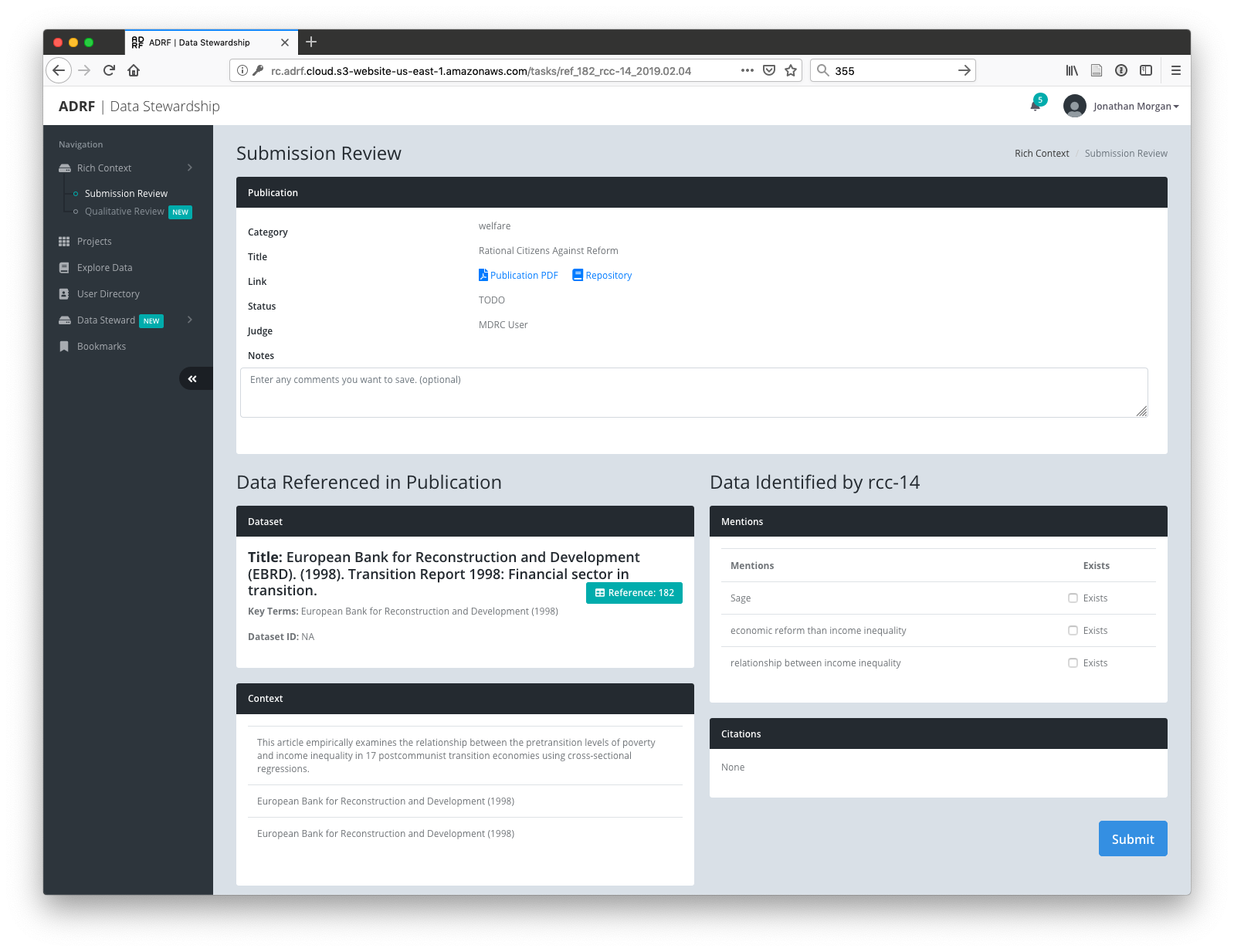
The coders, subject matter experts in the different key topic areas, looked at each “data reference” in publications in their area of expertise, evaluating it against the mentions and citations output by the model of each team that found data mentions or citations in the publication. For each reference-team pair, the coder flagged any mentions or citations “related to” the current data reference.

In our protocol (<https://docs.google.com/document/d/1Hi13N6gfiRz9nfwCoUQrey8v_ozY7fKHMtHV4GgX2ys/edit#>), we describe the coding task as “When you are judging data mentions, we want to mark mentions on the right as “exists” if they are related to the data referenced on the left, and make sure to not mark any mentions as "exists" that are not related.”, balanced with “If in doubt, don’t mark a given mention as related.”

The definition of “related to” is purposely fuzzy. Our goal was to give credit for finding language related to a dataset even if it wasn’t a perfect, formal reference, but to also make sure to not mark things that are obviously unrelated. To help to flesh this distinction out, we gave examples and analogies and training, and we had coders work through a few data references on their own then discuss their decisions.

An example from the protocol: “Think of it as a fuzzy match - we want to give the models the benefit of the doubt if they get close, especially if they detect some but not all key terms or phrases or find a mention of the basic type of data a named data set represents (“wage data” for IDES Unemployment Wage Records, for example), but we also want to make sure to reject things that are obviously not related.”

Coders used a web-based coding tool that listed out their assigned coding tasks and pulled together all of the information so they just had to scan the page, open the associated PDF if they had questions, and then mark related items and Submit to save their coding:



*Figure 4: The interface given to judges to evaluate whether a given team’s data set mentions and citations were related to a given data reference.*

As one would expect, while we got coders on the same page, each had subtly different ideas about what was or was not “related to”. To remove some of this variability from our final data, we then had a sole experienced researcher who understood what we were trying to do review all coding and, when he saw coding that obviously did not fit his understanding, either: revise to fit his understanding of “related to”; or flag as one he was unsure of and note his thoughts.

This experienced researcher also served as a final reviewer of the data references that were collected, marking any that did not actually refer to data as needing to be removed from our final analysis.

Finally, the protocol designer reviewed all removed data references, corrections, and ambiguities flagged for additional review, and made a final set of corrections.

#### Scoring the Results

To create a “related to” confusion matrix for each team, we started with a list of all of the data references that our final reviewers indicated should be included in our analysis (165 total). We created found-or-not (1 or 0) vectors with a value for every reference set to 1 for the baseline, and then set based on our coding for each team. For each publication, we also included a false positive item that was always 0 for the baseline, and that was set to 1 for a given team if they had any mentions or citations that were not “related to” a data reference from that publication.

To build a given team’s vector, for each data references, we checked to see if any of the team’s mentions or citations had been marked as “related to” that reference. If one or more of the team’s mentions or citations was marked as “related to”, we gave that reference a “1” for that team. If not, we gave it a “0”. Then, for each publication’s false positive item, if the team had 1 or more mentions and/or citations that were not “related to” any data reference, the team got a “1” for that entry. If not, they got a “0”.

To build out a confusion matrix, we went reference by reference: If the team found mentions and/or citations related to the reference, that was counted as a true positive (TP). If a team did not have any mentions or citations related to a given data reference, it was counted as a false negative (FN). Then, for the publication, if the team had 1 or more mentions and/or citations that were not “related to” any data reference, this was counted as a false positive (FP).

We did not develop a way to capture true negatives since the metrics we used to evaluate did not require it.

### Notes on Phase 2 Evaluation

This design was an attempt to move from just looking at titled data sets used for analysis and begin to look at all the ways data is discussed in academic papers, and how much of that discussion the combined mentions and citations of each team was aware of. Its effectiveness depended on how well we designed and carried out each of these three steps.

We are comfortable with the quality of the resulting data, but it should be noted that given the time and resources available to us, we had to make a choice between quality of data and reproducibility. In a perfect world, content analysis is the discipline of reliably being able to use a well-designed protocol to create content of comparable quality regardless of who does the coding. Given this project’s relatively tight timelines and limited resources, in this process we prioritized quality of data over reproducibility. We created relatively detailed coding protocols for each step of the process and we designed review and refinement into our processes, but we did not have time to go through multiple rounds of training and evaluation to make each of the protocols reliable and reusable. At the end, we introduced consistency by having experienced researchers familiar with our goals review all the output and either correct problems or flag items that should not be used in analysis. We assert that this created a reasonable level of consistency and quality in our output. We intend to refine these protocols for use in the future, however, and when we do, we will need to revise our collection methods substantially to make them more reusable and reproducible.

# Conclusion

TK