Gender Representation and Opinion Detection in the Media

Team Members: Jade Clarke, Lia Chin-Purcell, Kailin Koch, George McIntire, Siqi Wu
Advisor: Professor David Bamman
Organization Partner: Rethink Media

Table of Contents

Summary

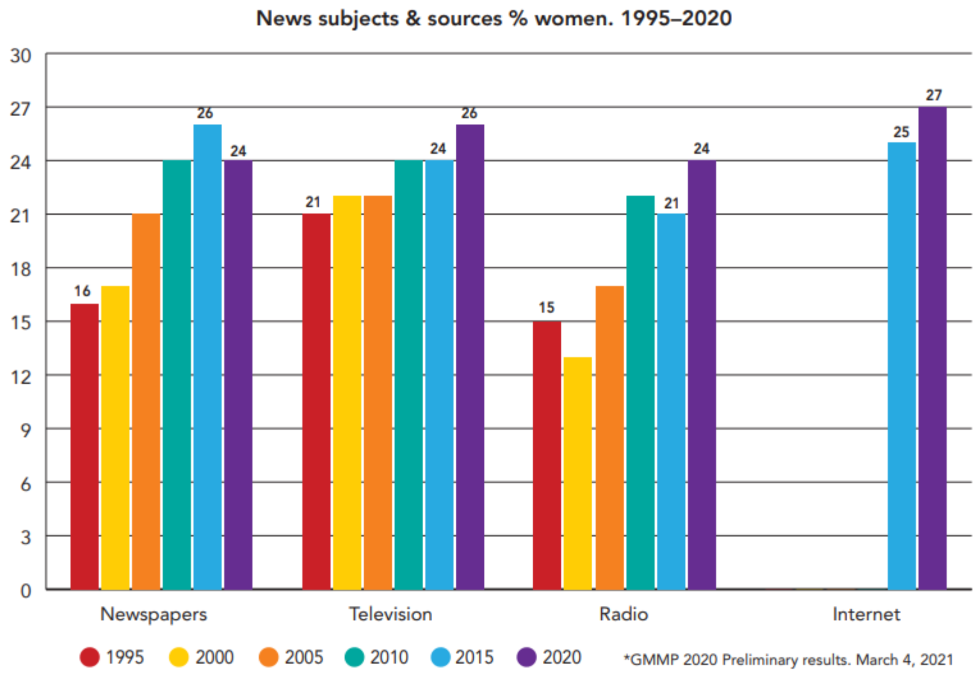
We built a dashboard for ReThink Media, a local Berkeley nonprofit, to help them visualize gender representation through news articles in their various issue areas. This builds off of previous work such as Informed Opinion's Gender Gap Tracker and the Global Media Monitoring Project's Who Makes the News Report. As part of this project we productionized the models, built a data pipeline, performed usability testing, and documented and handed off our work to the organization.

Helpful Links

- Github Repository
- Dashboard
- Project Page
- Website
- Documentation

Motivation - Current Female Representation in the Media

The Global Media Monitoring Project' Who Makes the News report found that as of 2020 in the 120 countries that were researched, the number of female news sources and subjects were 24% in newspapers, 24% in radio news, 26% in TV news. On the internet, the number of female news sources and subjects was 27% in 2020.



*GMMP 2020 Preliminary results. March 4, 2021

After COVID-19, news consumption shot up all over the world as society experienced their first global pandemic in almost a century. A new perspective on “news avoidance” emerged as the public had to be updated almost by the minute on crucial public health information, which has forever impacted the way we consume media. Outlets and social media platforms saw record numbers of new subscribers, and journalists were forced to be on the front lines more than ever. What still hasn’t changed is the gender disparity in news sourcing and voiceshare. Women now make up almost 50% of the workforce and still are marginally represented as reporting sources regardless of topic and geographic location. In the last 25 years, female representation has remained modest mostly due to awareness. Not enough people know this discrepancy even exists, forcing organizations like the Global Media Monitoring Project and ReThink Media to lead the fight to promote greater gender equality in voiceshare. By working with independent journalists and large omni-channel media outlets to alert them of this unequal balance, media will be forced to gradually make conscious efforts to include more women as their sources. We as a team hope to play a small part in helping ReThink Media acquire tools and resources to further their mission.

Motivation - Building the Models and Dashboard

Currently, ReThink media has a database of news articles that need to be manually classified as news/opinion articles. Additionally, people need to manually extract and attribute all quotes in the article. While there are existing tools available, they needed to be trained and tailored specifically on this news dataset. Additionally, typically these very technically involved and resource intensive processes. We set out to automate these steps in a simple, reproducible process.

Requirements

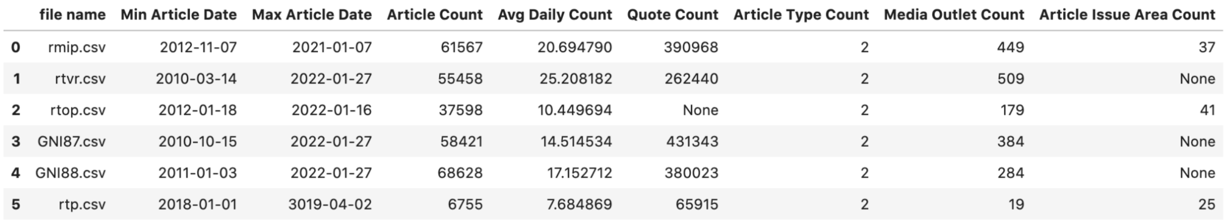
Below are some basic requirements for our final deliverable dashboard

- Publicly available dashboard displaying the rates of representation in the news by issue area over time.
    - MVP: nuclear issue area only
- Classify articles as news vs opinion programmatically rather than by hand
- Extract quotes and speakers programmatically, replacing the current manual process
- Connect to existing gender inference pipeline
- Manually review data ahead of publishing new data
- Easily update model logic and troubleshoot

Background Research

To begin, we conducted 2 literature reviews. The first focused on existing methods for news article classification, gender classification and quote extraction. This document can be found here. Secondly, we surveyed research about how women are represented in the news media, including both their characterization and the substance of their coverage. This document can be found here. In addition, we researched a number of publicly available tools we could leverage for this project, including github pages, Google Data Studio, Tableau Public and Amazon Web Services. Lastly, we spoke with the client extensively about their existing tooling and reached out to a former iSchool capstone team to understand why they had chosen specific tools.

We also explored ReThink Media's existing data for this project. Below is a summary of the data files ReThink already has for news articles. Our MVP launch focuses on data in GNI88.csv, which focus on nuclear issues.



	file name	Min Article Date	Max Article Date	Article Count	Avg Daily Count	Quote Count	Article Type Count	Media Outlet Count	Article Issue Area Count
0	rmip.csv	2012-11-07	2021-01-07	61567	20.694790	390968	2	449	37
1	rtvr.csv	2010-03-14	2022-01-27	55458	25.208182	262440	2	509	None
2	rtop.csv	2012-01-18	2022-01-16	37598	10.449694	None	2	179	41
3	GNI87.csv	2010-10-15	2022-01-27	58421	14.514534	431343	2	384	None
4	GNI88.csv	2011-01-03	2022-01-27	68628	17.152712	380023	2	284	None
5	rtp.csv	2018-01-01	3019-04-02	6755	7.684869	65915	2	19	25

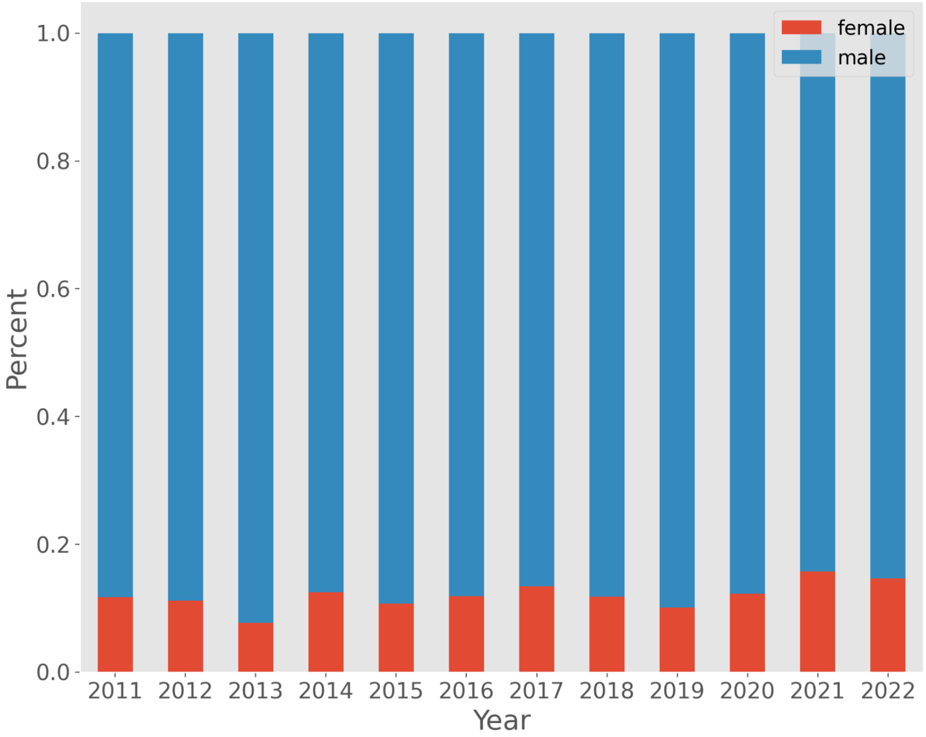
Analysis

After building our NLP pipeline, we used it to process over 60,000 articles. Given the wealth of data derived from the corpus using our tools we analyzed the data to search insightful patterns illuminating how men and women are quoted in this particular section of the media.

Some of our findings include:

- Around 75% of news articles do not quote a single woman.
- The ratio of male quotes to female is 8:1
- 15% of journalists with at least 5 articles quote zero women.

- Average article has 3.68 quoted men vs 0.8 quoted women.
- We found this ratio to be consistent when analyzing different groups of outlets i.e. local vs national and magazine vs newspaper vs radio vs online.
- Outlets with the highest rates of female quoting typically had about 20% of their quotes come from women.
- We detected no significant difference in the length and content of quotes from men and women.
    - Both uses 18 words on average in their quotes
    - Using topic modeling, we observed little in the topics scores when grouping by gender.
- The rate of female quotes has stayed consistent over the years, see the following chart.



Natural Language Processing Models

Owners: Lia Chin-Purcell & George McIntire

Opinion Classification

Goals

- Classify the incoming flow of news articles as either “opinion” or “news.”
- Serves purpose ensuring the ensuing models are applied to “news” articles.
- This is the first step in the modeling process, the articles classified as “news” will then be passed onto the next step in the modeling process.

We used a corpus of 6342 news articles provided to us by ReThink pre-labeled as “Opinion” or “News” — the two classes are of approximately equal proportions. The opinion classifier is an ensemble model comprised of a MLP neural network trained on the sentence embeddings derived using the [paraphrase-MiniLM-L6-v2]() model and a DistilBert base cased model fine-tuned on our corpus. Our only preprocessing step involves using the python package [Texthero]() to clean the raw article text.

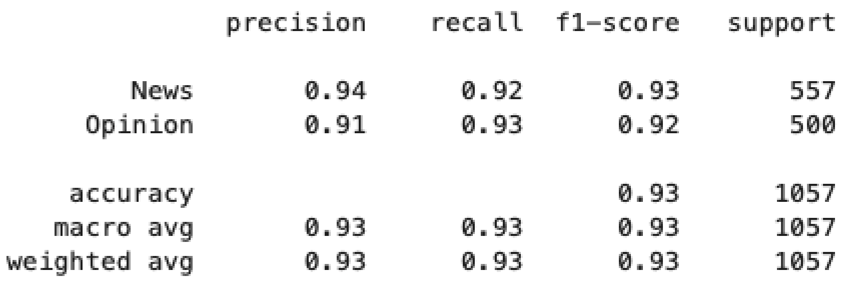
The opinion classification process follows these steps:

1. If an article happens contains phrases or words such as “Opinion”, “OP-ED”, or “Editorial” then it is automatically classified as opinion.
2. Pass the raw text into the Texthero text cleaning pipeline.
3.
    a. Derive sentence embeddings from the processed text.
    b. Tokenize processed text using the Distilbert’s tokenizer.
4.
    a. Use each model to derive the probability of opinion.
    b. Average the two probabilities and if that value is greater than 0.5 or a desired threshold then classify article as opinion or otherwise send article onto the quote extraction and gender inference step.

Evaluation

After an exhaustive process that involved testing and experimenting a variety of text classification and processing methodologies, our method listed above produced the best overall results.

The following displays the performance of our modeling on the validation data using a threshold of 0.5



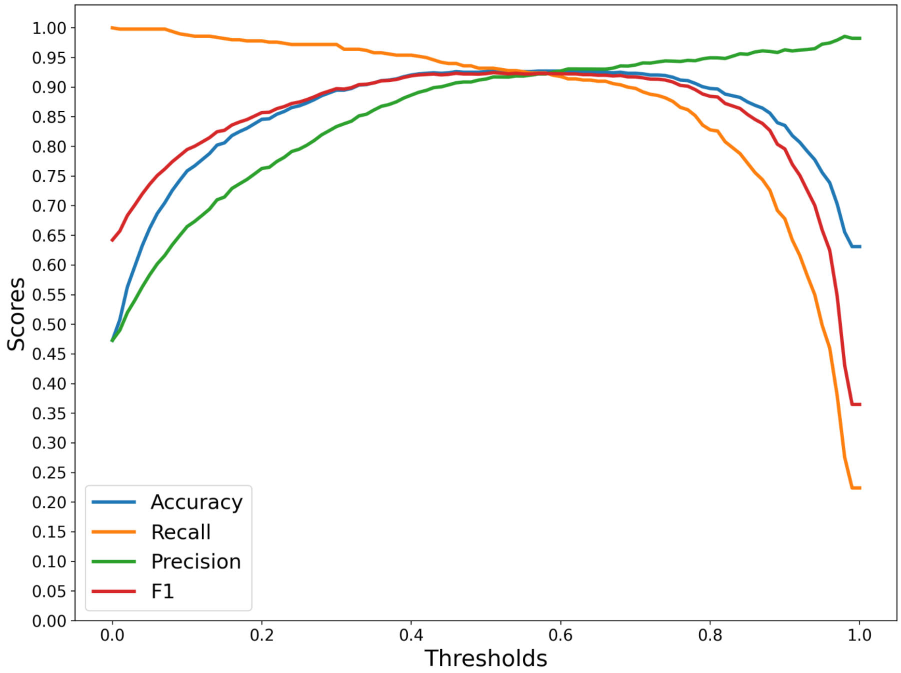
```
              precision    recall  f1-score   support

        News       0.94      0.92      0.93       557
     Opinion       0.91      0.93      0.92       500

    accuracy                           0.93      1057
   macro avg       0.93      0.93      0.93      1057
weighted avg       0.93      0.93      0.93      1057
```

Our model’s performance indicates that our methodology can be expected to correctly classify 93% of opinion articles as opinion and 92% of news articles as news.

Setting a threshold for opinion classification has been a recurring topic for discussion and we’ve not settled on a precise value. It may be recommended to choose a lower threshold in the range of 30-45% in order to filter out an increased number of opinion articles while still retaining almost 90% of news articles. We did analyze the relationship between model performance and various thresholds. The following plot displays validation scores for accuracy, recall, precision, and f1 for a range of thresholds from 0.0 to 1.0 — opinion is considered the positive class in this example.



Quote Extraction and Attribution

Goals

- Extract quotations in news article text. Specifically, for each direct and indirect quotation in an article, identify the quote content, the verb cue, and the speaker
- Evaluate the degree to which the model is able to extract all of the quotations present in an article, and how many the model is falsely identifying as a quotation. I.e. evaluate the accuracy, precision and recall of the model

The quote extraction model takes in a document, or the text body of an article, and returns a list of the direct and indirect quotations that appear in the article. For each quotation found in the article, the quote extraction model identifies the quote itself, the speaker of the quote, and the verb cue (the verb indicating that a quotation is there such as “says”, “said” etc). For example, from the follow piece of text:

> Russia quietly boosted economic support for North Korea this year, and last week Russian Deputy Foreign Minister Igor Morgulov said Moscow was not ready to sign up to sanctions that would strangle the country economically.

The quote text is “Moscow was not ready to sign up to sanctions that would strangle the country economically.”, the speaker is “Russian Deputy Foreign Minister Igor Morgulov”, and the verb cue is “said”.

We use and build upon Toriabi Asr et al.’s quote extraction model that powers The Gender Gap Tracker dashboard. In this model, we use the spaCy library to parse text and identify linguistic features of the text, in particular the syntactic structure of sentences. By identifying the part of speech of different words and their linguistic dependencies, the model identifies potential quotes by checking if the verb cue is in a list of potential verb cues, and if so, extracts the corresponding quote and speaker. Within this general method, there are several functions in the model: one that extracts syntactic quotes like the one mentioned above, and one that extracts floating quotes, and a heuristic method that extracts syntactic quotes that are missed. An example of a floating quote is found below:

> “Honestly, it feels like we’re living our worst nightmare right now” Kim told CTV News Friday. “The fact that we are being accused right now of an unethical adoption is crazy.”

Here, as readers we know that the second quote “The fact that we are being accused right now of an unethical adoption is crazy.” the speaker is also Kim, which the model is able to capture with an additional function.

The model also extracts heuristic quotes, or quotes that are enclosed between start and end quotation marks. The model then identifies the speaker of the quote by identifying the closest subject associated with the verb cue with the linguistic dependencies identified by the spaCy model.

Evaluation

For evaluation, we hand-annotated 20 articles for their quotations. To measure the accuracy, precision and recall of the model, we use the same method as Toriabi Asr et al. and identify that the model has extracted the “same” quote that a person has annotated based on a threshold of overlap. This threshold can either be specified based on the indices of the characters within the article body, or as a fuzzy matching ratio threshold.

For the fuzzy matching threshold, we use the fuzzywuzzy package to calculate the Levenshtein distance ratio of similarity between two quotations. At a 60% ratio threshold, we obtain the following metrics:

Precision	69.0%
Recall	77.0%
F1 score	72.8%

These were our final metrics after iterating on the algorithm several times, including adjustment to thresholds, stop-word lists, and data pre-processing.

Gender Inference

Goals

- Cluster the ‘person’ entities in each article through coreference resolution, and associate quotations with entities
- Infer the gender of person entities, so we can determine quotes spoken by men and quotes spoken by women

To identify the person entities in the article and their names, we use named entity recognition and coreference resolution functionality provided by spaCy. To use this functionality, we add a neural coreference model to the spaCy pipeline. This coreference model works well with our purposes and model, but there are more sophisticated models that could be used for this task in the future.

After identifying the entities and coerferring them to each other, we filter on speakers who have been identified with the ‘person’ entity, and extract the name from the coreferring cluster. After extracting the names of the speakers who are people entities in an article, we pass the names to our gender inference model, which is essentially a series of calls to various gender inference APIs.

The first step of the model is to check if the name and its corresponding gender appears in our database of cached names. The cached names come from previous API calls. If the name does not appear in the database, we then get the gender from the library gender guesser. This library tends to give a lot of ‘unknown’ and ‘andy’ (indicating androgynous), in which case we then make an api call to gender api. If gender api also gives unknown, we then make an api call to genderize. If the name is still unknown at this point, we infer the gender as ‘unknown’.

Once we have inferred gender for all of the speakers of the articles, we can then associate each quote spoken by a person to the name of the speaker and their inferred gender. It is important to note that currently, the model relies on various calls to gender APIs, which, particularly for the gender guesser model, are typically trained on predominantly western names, and is thus less accurate for non-western names. However, GenderAPI performs well on non-western names, with an accuracy drop-off of only around 1-2% for non-European names. Depending on which API is called, this bias would be more or less prevalent. This is an area of unfairness in the model. Another area of unfairness is that the model currently only outputs woman, woman and unknown, which does not capture the nuance of gender as mutable and non-binary.

Evaluation

To evaluate the coreference resolution, we evaluated hand-annotated names of speakers from all of the quotations in an article, against the extracted speakers from our model. This method uses fuzzy matching to ensure a name match, and gives accuracy, precision and recall metrics.

To evaluate the accuracy of the gender inference we relied on the accuracy reporting already completed by ReThink media for gender inference. This involves using gold data for speaker names and their genders which was compiled using online databases, and standard measures of accuracy, recall and precision for a binary classification task. On the GNI88 dataset, the GenderAPI model is 95.7% accurate with unknowns, and 96.2% accurate without unknowns.

Potential Future Work

As touched on above, one area of future work for the entity resolution and gender inference step is to incorporate a more sophisticated model for the coreference resolution. Specifically, AllenNLP has an end-to-end coreference model that could be used for this task. Additionally, we could use pronouns within a coreferring cluster to infer gender for those who use they/them/theirs pronouns.

Additionally, The Gender Gap Tracker's codebase also includes the possibility to capture and infer the gender of the author of the article, so incorporating inferring the gender of the author into the modeling would be a natural next step. Since the author name is already included on the quote level, inferring their gender would simply imply passing it to the gender inference function, and including their gender as a new field.

Data Engineering

Owners: Siqi Wu & Kailin Koch

Goals

The overarching goals of this design were as follows:

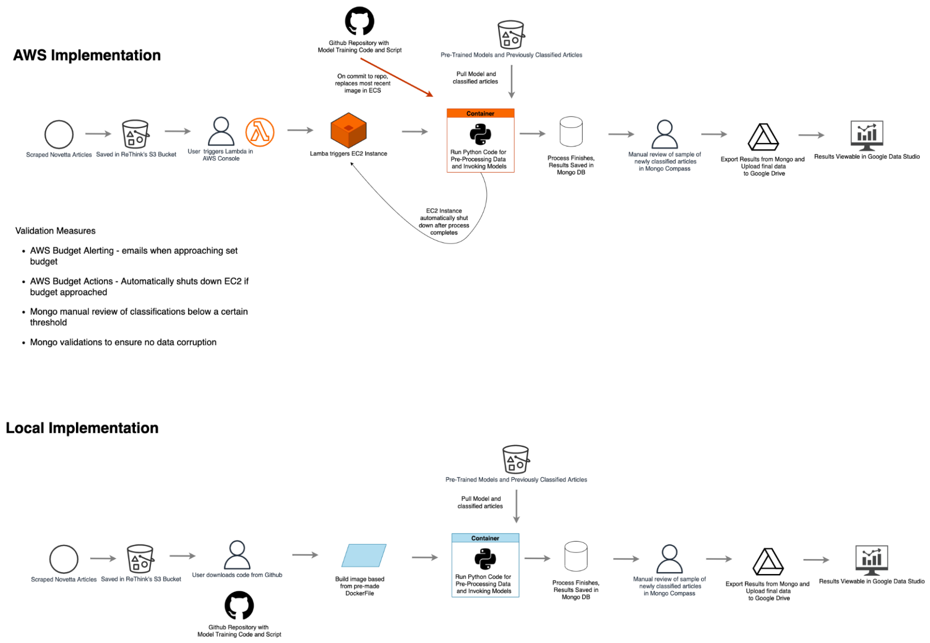
- Consolidate tooling as much as possible to tools the organization is familiar with and uses currently
- Automate as much of the pipeline as possible to make it easy to run and maintain
- Utilize low cost and free services wherever possible, given we are designing for a non-profit organization with limited resources
- Make it easy for the team to update the preprocessing and model code without having to make changes to the data pipeline
- Connect the entire pipeline from data ingestion to local display

Approach

Given our design goals, we opted to build this pipeline using Amazon Web Services (AWS), MongoDB, and Google Data Studio. All three tools are currently in use by the organization today, are easy to update and maintain, and are free or low cost. Within AWS, we experimented with different services and landed on services that could be both relatively low costs and provide enough computation power. We considered using Tableau Public in lieu of Google Data Studio, but it offered fewer permissioning options and would've been an additional tool many stakeholders had less experience with. With all these tools, we automate the process as much as possible so our clients could use the pipeline without knowledge in building docker images or writing command lines.

Design

A diagram of our current data pipeline is below.


AWS Implementation
Github Repository with Model Training Code and Script
On commit to repo, replaces most recent image in ECS
Pre-Trained Models and Previously Classified Articles
Pull Model and classified articles
Scraped Novetta Articles
Saved in ReThink's S3 Bucket
User triggers Lambda in AWS Console
Lamba triggers EC2 Instance
Container
Run Python Code for Pre-Processing Data and Invoking Models
Process Finishes, Results Saved in Mongo DB
Manual review of sample of newly classified articles in Mongo Compass
Export Results from Mongo and Upload final data to Google Drive
Results Viewable in Google Data Studio
EC2 Instance automatically shut down after process completes
Local Implementation
Pre-Trained Models and Previously Classified Articles
Pull Model and classified articles
Scraped Novetta Articles
Saved in ReThink's S3 Bucket
User downloads code from Github
Github Repository with Model Training Code and Script
Build image based from pre-made DockerFile
Container
Run Python Code for Pre-Processing Data and Invoking Models
Process Finishes, Results Saved in Mongo DB
Manual review of sample of newly classified articles in Mongo Compass
Export Results from Mongo and Upload final data to Google Drive
Results Viewable in Google Data Studio

Validation Measures

- AWS Budget Alerting - emails when approaching set budget
- AWS Budget Actions - Automatically shuts down EC2 if budget approached
- Mongo manual review of classifications below a certain threshold
- Mongo validations to ensure no data corruption

Data Cleaning

Before running the models on the data, we cleaned up the scraped articles by removing some metadata from the article body. To limit the amount of data we need to run the models on, we check whether we've already classified each article prior to running the pipeline. This way the pipeline only checks new articles.

Running Models in AWS

Given the time and size of the models, we built a system on AWS so that the client could run the models without worrying about needing to monitor it too carefully. After manually triggering the job in AWS, a lambda function will start an EC2 instance. On this EC2 instance, we will run the pipeline code. This includes getting new articles, cleaning, running the models and saving the results. Once this process is complete, the lambda function will shut off the instance again.

Running Models Locally

Because AWS can be costly and more technically involved, we also ensured the pipeline could be run successfully locally and included extensive documentation about how this could be accomplished. The client can clone the github repository, build a docker image and run it on their local machine. While slow and memory intensive, it provides a workable alternative to AWS.

Saving Results

Results are saved to a MongoDB database. From here, the team can run data validations, aggregate the collected quotes and articles, and export the results. MongoDB also has a simple GUI called MongoDB Compass, so it's easy to connect locally and interact with the collections. The client is familiar with MongoDB.

Throughout this project we often wrestled large amounts of data and free services with storage limits. This was especially true with MongoDB, which has a limit of 512MB for its free services. To be as space efficient as possible, we saved articles and quotes as 2 separate collections. This way, we didn't have duplicative information about the articles for each quote. The two collections could be easily joined (or in MongoDB speak $lookup) together on an article_id key.

Manual Review

We created 4 data aggregations in MongoDB so the client could manually review any classifications that we had less confidence in. The aggregations find articles that have not yet been reviewed that meet a certain criteria, export these results to a small collection to be reviewed, and then combine the results. We also do some light data cleaning ahead of displaying the results in the dashboard. (See Appendix for full list of rules)

Updating Dashboard Data

Once the manual review is complete, the data can be downloaded and loaded into Google Data Studio. Given the limitations of file sizes allowed as CSVs or Google Sheets, we limit the imported data to only the relevant information displayed in the report. We also build a pivot table in Google Sheets to power the 'comparison of voiceshare' chart on page 4 of the report.

Operationalizing The Design

Integrations

- **Storing Models -** the models are saved in Github and Google Drive. They are pre trained and called in the code.
- **Getting New Data -** to avoid duplicative work, we perform a check on the articles in the S3 bucket to only clean and classify the new articles.
- **Saving to MongoDB -** after the results are run, they are saved in a MongoDB database. This makes it easy for the client to review the results, run data validations and aggregations.

- **Automating Building Docker Image** (Github > AWS) - we created a GitHub action which automatically uploads a new Docker Image to the AWS repository every time new code is pushed in the github repository. This saves the client the hassle of making sure the Github repository and AWS account stay in sync with the same DockerFile and image.

Managing Costs

- **Budget Alerts** - there are multiple alerts tied to AWS spending to ensure the client has no surprise charges. These alerts go to multiple members of the organization and to position specific emails, to ensure they reach their intended recipient even if specific individuals change roles.
- **Budget Actions** - after the budget alert has been reached, we have configured a budget action in AWS which will shut down the EC2 instances without needing human intervention. This will prevent the EC2 from racking up additional charges. However, this alert must be tied to specific EC2 instances, and so it must be updated if a new EC2 instance is created. This is documented in the client documentation.
- **Lifecycle Policy** - to ensure that we don’t unnecessarily store old docker images in AWS, we’ve implemented a lifecycle policy on the repository. It will only keep the newest image in the repository.

Managing Performance Issues

- **Running the Models** - If there are large amounts of new articles to process, the model will take a long time to finish. To track if the model is still running, we built in a logging mechanism where a status log will be written into a text file after each step (i.e. data cleaning, opinion classification, and gender identification). This text file will be uploaded to S3 immediately after each update. As such, users can download the log from S3 and check in the status of the pipeline.
- **MongoDB** - We stored the results as two separate tables to avoid storing duplicative information. Thus we had an articles table and a quotes table, and the quotes could be joined to the articles. This sped up the queries of both and it was trivial to join them together.
- **Google Data Studio** - To avoid lagginess in the dashboard, we created a data extract which makes a local copy of the data to avoid a live connection. This extract updates automatically from the source every day.

Reflections

How well did we achieve our objectives?
What unforeseen obstacles did we hit and how did we overcome them?

Overall, we accomplished what we set out to achieve with the data pipeline. We underestimated how long it would take to run the gender inference pipeline and how much the size of the data would impact our pipeline. For example, we hit the data limits in the MongoDB free tier and the maximum size of a Google Sheet. As such we had to adjust how we organized the data. For example, we stored the data as two separate tables, one which contained the article classification and the other which contained the quote and gender classification. That way, we didn’t have to save duplicative copies of the article for each quote and could instead join the quotes on the article ID.

Potential Future Work

For future iterations of the pipeline, we think the opportunity areas are improving latency and making the github repository more secure. Currently, running the full pipeline on the decade of news articles takes multiple days and costs a few hundred dollars to run. Running the models more quickly will decrease iteration speed and allow for more improvements to the models. 2 approaches to try are parallelization and AWS Batch. AWS Batch has been found to run faster than EC2 for some customers and is specifically built for training and running ML models. Optimizing the quote extraction and gender inference models might also help speed up this stage of the pipeline. Additionally, splitting up the manual review in 2 and reviewing the news classification and then only running the quote extraction on the news articles, not opinion would mean less data and thus less time. However, this might be more of a headache for the users.

Currently, the GitHub repository contains the access information for the AWS account in a password protected CSV file. It would be good to eventually move this to a github secret or another more secure solution, particularly before this repository is made public.

UX Design and Research

Owner: Jade Clarke

Overall Goals

The overall goals for this project in terms of UX and Design were as follows:

- Become an advocate for the end user: ReThink Media employees
- Conduct research to understand major pain points and functionalities
- Design the dashboard to include optimal functionality
- Conduct usability studies to test the user friendliness and usability of the dashboard

Stakeholder Interviews

Stakeholders are people who have stakes in the project. They are people who make decisions regarding a UX design project. The stakeholders are the owners of the product and can range from founders, executives to project managers. In this project, our stakeholders are ReThink Senior Leadership, Program Managers and Research Analysts. And hence a stakeholder interview is a consequential step that can't be avoided.

I began by utilizing our ReThink contact to connect me with employees who she believed would be the most valuable to gain insights from on what the dashboard should accomplish for the company. After compiling a list of interview goals, I created an interview protocol covering eight topics ranging from project vision, business goals, users, proposed functionality and technological limitations.

I conducted 4 stakeholder interviews over a two week period, each lasting 45 minutes to an hour. After conducting the interviews, I compiled all the findings into reports and presented them to my capstone team and Laura Nixon. Using these findings, I was able to compartmentalize what the dashboard should be able to do which in turn informed my usability studies and the aesthetics of the dashboard.

Findings

Some major highlights synthesized from the stakeholder interviews:

1. The dashboard should be able to be shared with partner organizations
2. The dashboard should have variability in sharing and viewing capabilities
3. Data sensitivity should be a main priority
4. Dashboard should be easy to maintain and update should new data come in
5. Dashboard should provide a robust landscape of the current female voiceshare in US News and Media
6. Dashboard needs to come with documentation for the purposes of troubleshooting and continuity after the team graduates

Metrics to be included:

- Breakdown of quotes by news source
- Breakdown by topic
- Volume data over time
- Ability to layer in gender breakdown and news outlet
- Filter by name
- Filter by outlet
- Filter by unique identifier
- Filter by total volume
- Being able to create charts (line graphs + bar charts) and export them as images
- Show percentages in relevant areas
- Have annotations
- Creating visualizations such as charts

Usability Tests

What is usability testing?

Usability testing is a method of evaluating a website's or app's readiness for release by testing it with real users who are part of the target audience. In our case, we are evaluating the dashboard. Usability tests evaluate the overall user experience by measuring the relative ease with which end users can accomplish a set of tasks that a typical user of the dashboard would need to accomplish.

Why is it important?

The goal of usability testing is to understand how real users interact with our dashboard and make changes based on the results. It is important to be sure that the dashboard is easy to navigate and that tasks can be completed with ease to meet the needs of ReThink employees like you.

I conducted usability studies over the month of April to test the functionality of the dashboard. I wanted to get a range of technical and non-technical ReThink employees. To reflect the average user of the dashboard, employees who work with data and who do not are both equally important as all employees at ReThink will have use for this dashboard. I conducted two rounds of usability tests with four participants in order to gain more breath on how user friendly the dashboard should be. It is more valuable to conduct multiple rounds with a smaller population than conducting one round with a larger population in order to maintain continuity. An interview protocol was written that included goals, optimal user profiles and scenarios and tasks that reflect a day to day life of a ReThink employee.

In each usability study, participants conducted at least four tasks virtually over zoom testing various functionalities that were informed by the stakeholder interviews. The interviews were conducted over zoom with the participants sharing their screen with the moderator to complete the tasks. The dashboard was being tested with a small subset of the full data in order to streamline time limitations. Tasks ranged from utilizing certain filters to populate visualizations, exporting documents from the dashboard, and interpreting data. After the first round, major concerns were taken to the team to make changes and then those changes were tested in round two. Round one took place from April 11th to April 18th. Round two took place April 19th to April 25th. These interviews were recorded and compiled into findings reports as the stakeholder interviews were.

Findings

After Round One:

Positive Comments:

- Dashboard is robust
- Easy to use
- Aesthetically pleasing
- Visualizations are easy to populate
- Visualizations are relevant to dashboard MVP
- Filters are simple to use for even non-technical users

Areas of Concern:

- Desire a way to compare certain metrics to each other
  - Journalists
  - Media Outlets
- Modify the drilldown from Month and Year to just by Year

After round one, we made two major changes to the dashboard. We added a 5th page to the dashboard entitled “Comparison of Female Voiceshare.” This page gives users the ability to compare Media Outlets to each other and visualizes the percentage of female voiceshare in selected Media Outlets over time. The second major change was changing the chart type in our "Over Time" page. The chart was changed to a 100% stacked bar chart in order to make visualizing trends over time easier for the user.

After Round Two:

Positive Comments:

- Comparison of Female Voiceshare was a great addition
    - Easy to interpret
    - Liked the ability to compare Media Outlets
    - Highlighting female voiceshare in the chart is great for ReThink's overall mission
- New stacked bar chart made it easier to display trends over time

Areas of Concern:

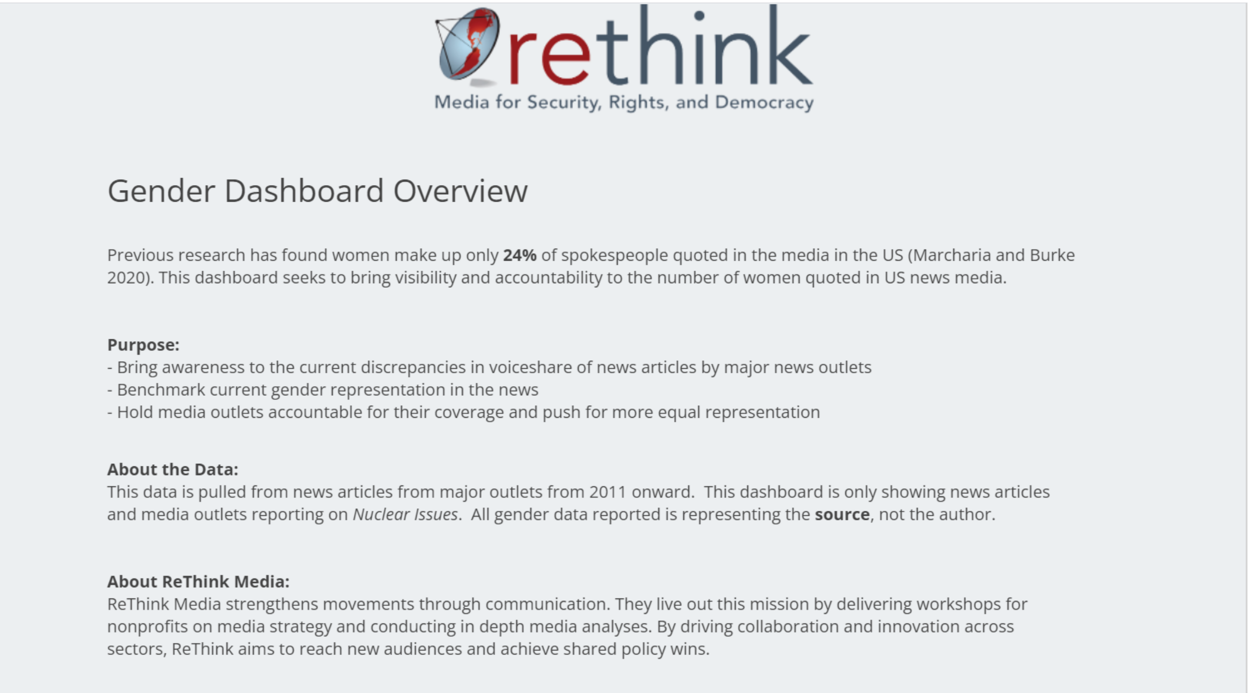
- Add a sentence on each page of the dashboard as an introduction to what the page's function is
- Modify some information on the Summary page to reflect 2022 statistics on Female Voiceshare

Dashboard Creation

Our dashboard was created utilizing **Google Data Studio**. Google Data Studio is an online tool for converting data into customizable informative reports and dashboards. We chose this tool because of its familiarity within G Suite for the workforce and its easy ability to change sharing, editing and viewing permissions to keep the sensitive data in the dashboard.

Our dashboard consists of 4 pages. Page 1 is a summary of the dashboard's purpose and MVP (Minimum Viable Product). Page 2 is a Totals page that displays the overall total percentages of male vs. female voiceshare in US News and Media. Page 3 is an Over Time page that displays female voiceshare over the time period of 2011 to the present. Page 4 is a Comparison of Female Voiceshare page that allows users to compare Media Outlets to each other and highlights the percentage of female voiceshare in each Media Outlet.

Each page in our dashboard contains the same four filters to populate visualizations. "Issue Area" refers to which ReThink issue area the data is populated for. For the purposes of this capstone project, we are working with the Nuclear Issues area but added the filter in case ReThink wanted to create subsequent dashboards for other issue areas in the future. There is a Date filter which allows users to pick the time frame they want to splice the data by. "Media Name" allows users to filter by certain Media Outlets that are in our database. "Journalist Name" allows users to filter by particular journalists to see their individual voiceshare within the Nuclear Issues issue area.



rethink

Media for Security, Rights, and Democracy

Gender Dashboard Overview

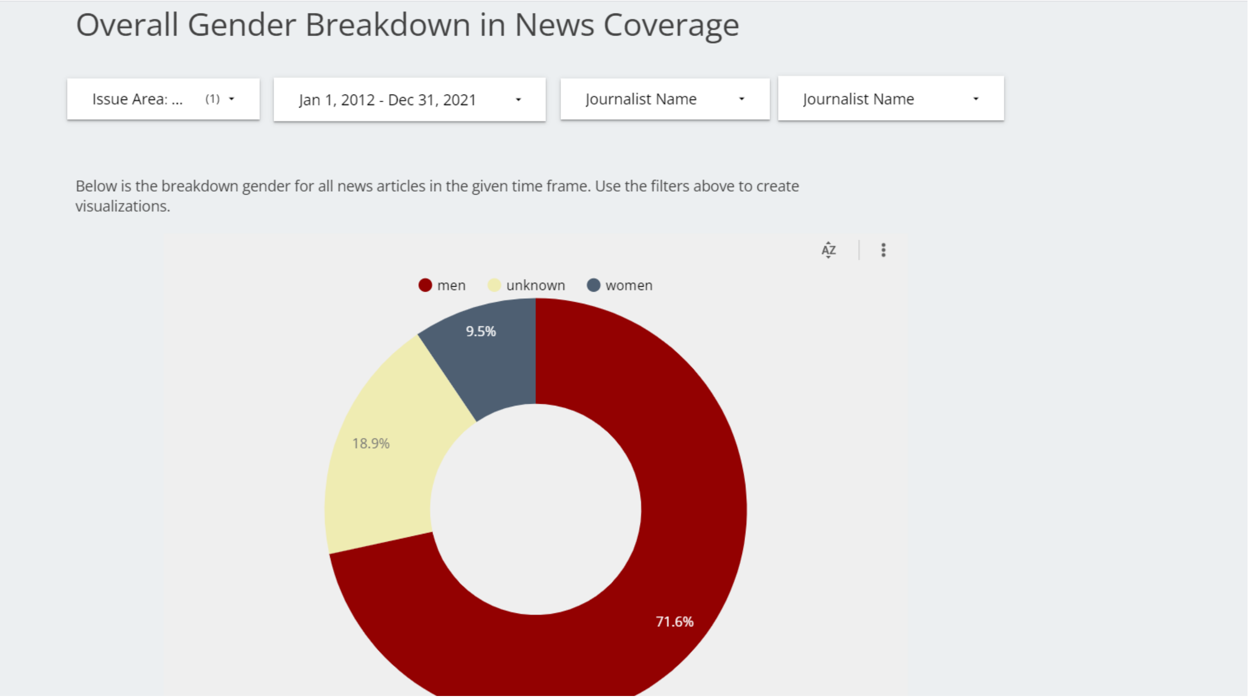
Previous research has found women make up only **24%** of spokespeople quoted in the media in the US (Marcharia and Burke 2020). This dashboard seeks to bring visibility and accountability to the number of women quoted in US news media.

Purpose:
- Bring awareness to the current discrepancies in voiceshare of news articles by major news outlets
- Benchmark current gender representation in the news
- Hold media outlets accountable for their coverage and push for more equal representation

About the Data:
This data is pulled from news articles from major outlets from 2011 onward. This dashboard is only showing news articles and media outlets reporting on *Nuclear Issues*. All gender data reported is representing the **source**, not the author.

About ReThink Media:
ReThink Media strengthens movements through communication. They live out this mission by delivering workshops for nonprofits on media strategy and conducting in depth media analyses. By driving collaboration and innovation across sectors, ReThink aims to reach new audiences and achieve shared policy wins.

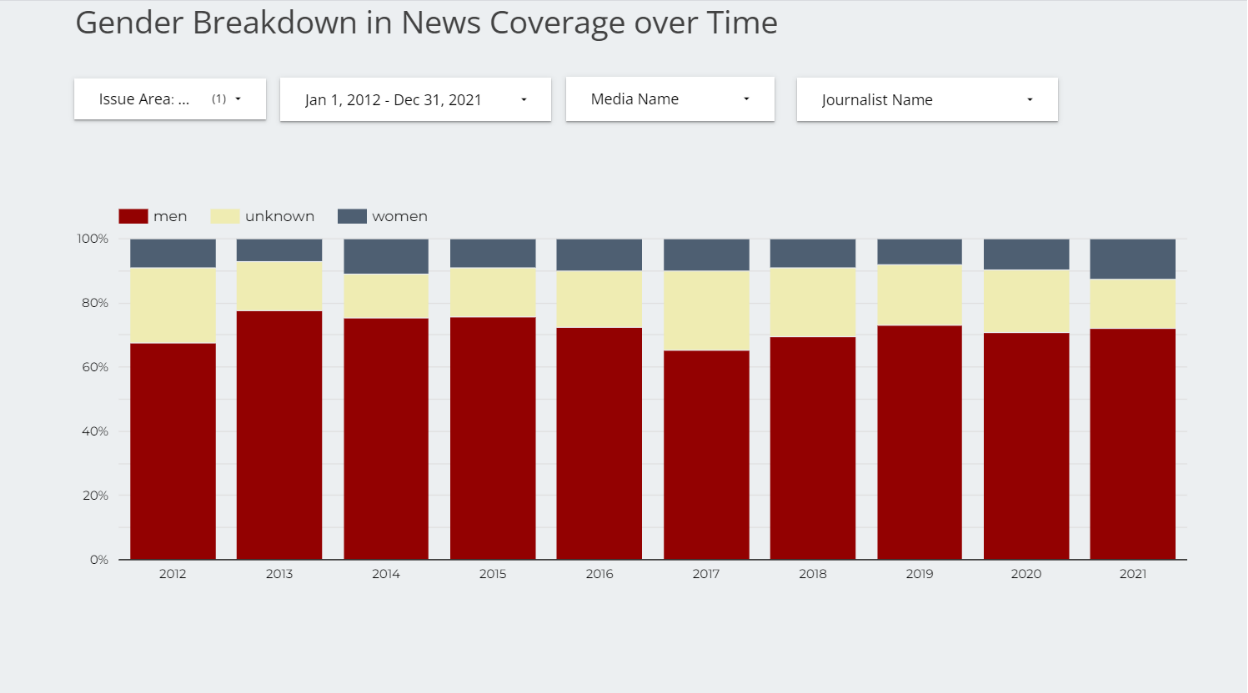
Summary Page



Overall Gender Breakdown in News Coverage

Below is the breakdown gender for all news articles in the given time frame. Use the filters above to create visualizations.

Totals Page


Gender Breakdown in News Coverage over Time

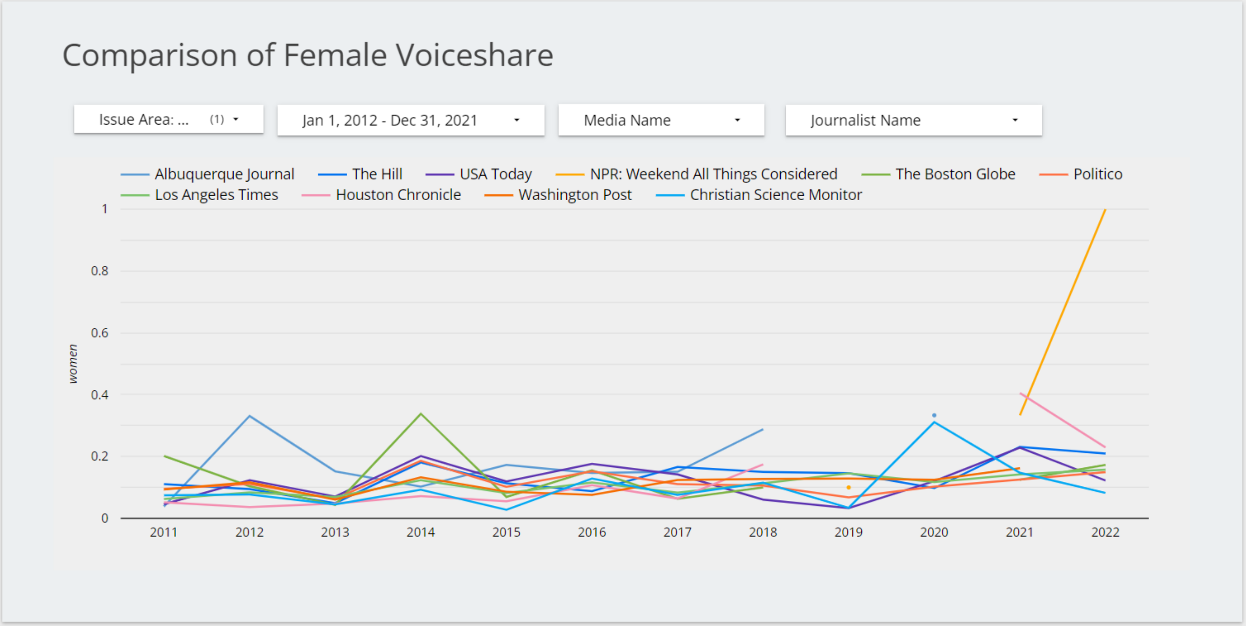
Issue Area: ... (1)

Jan 1, 2012 - Dec 31, 2021

Media Name

Journalist Name

Over Time Page


Comparison of Female Voiceshare

Issue Area: ... (1)

Jan 1, 2012 - Dec 31, 2021

Media Name

Journalist Name

Comparison of Female Voiceshare Page

The dashboard's aesthetics were created based on the ReThink Style Guide. I used their font choices (Open Sans) and primary company colors (Slate #4e5f72, Wheat #efecb2 and Crimson #930101) to make the dashboard flush with the rest of ReThink's materials.

Appendix

MongoDB Rules from Documentation

Data Validations

{"prediction": {$in: ["News", "Opinion"]}}

{"gender_corrected": {$in: ['men', 'women', 'unknown']}}

Make Article ID a unique key in results opinion

Data Aggregations

Step	Description	Collection	Output	Code	Notes

1	Gather News Articles to Manually Review	results_opinion	news_review	[{$set: {
				 prediction_final: '$prediction'
				}}, {$match: {
				 manual_review_news: {
				  $exists: false
				 },
				 probability: {
				  $lte: 0.01
				 }
				}}, {$addFields: {
				 manual_review_news: true,
				 manual_review_news_date: Date()
				}}, {$limit: 5}, {$out: 'news_review'}]
					Currently there is a limit of 5 articles on the exported file.

2	Save Manual Edits back to results_opinion collection	news_review	NA	[{$merge: { into: 'results_opinion', on: '_id', whenMatched: 'merge' }}]	

3	Gather quotes to Manually Review	results_quote_with_gender	gender_review	[{$addFields: {
				 gender_recode: {
				  $cond: {
				   'if': {
				    $eq: [
				     '$gender',
				     'male'
				    ]
				   },
					Warning: some values in gender probability are between 0 - 1, others 0-100. You can handle by adding an if condition
					{
					    $or : [{gender_prob: {$gte:1, $lte:20}},{gender_prob:{$

				then: 'men',
				'else': 'unknown'
				}
				}
				}}, {$addFields: {
				gender_final: '$gender_recode'
				}}, {$match: {
				gender_prob: {
				$eq: 0
				},
				manual_review_gender: {
				$exists: false
				}
				}}, {$limit: 5}, {$addFields: {
				manual_review_gender: true,
				manual_review_gender_date: Date()
				}}, {$out: 'gender_review'}]
					gte:0.1,$lte:0.9}}]
					}

4	Save Manual edits back to results_quote_with_gender collection	gender_review	NA	[{
				$merge: {
				into: 'results_quote_with_gender',
				on: '_id',
				whenMatched: 'merge'
				}
				}]	

5	Join quotes and articles, create final gender and news fields, export	results_quote_with_gender	final_results	[{$lookup: {
				from: 'results_opinion',
				localField: 'article_id',
				foreignField: 'Article ID',
				as: 'article'
				}}, {$set: {
				gender_recode: {
					Currently limit of 5 set on collection

```
  $cond: {
   'if': {
    $eq: [
     '$gender',
     'male'
    ]
   },
   then: 'men',
   'else': 'unknown'
  }
 }
}}, {$unwind: {
 path: '$article'
}}, {$set: {
 'Published Date': '$article.Published Date',
 'Journalist Name': '$article.Journalist Name',
 Headline: '$article.Headline',
 'Media Name': '$article.Media Name',
 prediction: '$article.prediction',
 prediction_final: '$article.prediction_final',
 probability: '$article.probability'
}}, {$project: {
 prediction_final: {
  $ifNull: [
   '$prediction_final',
   '$prediction'
  ]
 },
 'Published Date': true,
 'Journalist Name': true,
 'Media Name': true,
 id: true,
 article_id: true,
 gender_final: {
```

				$ifNull: [
				'$gender_final',
				'$gender_recode'
				]
				},
				'Issue Area': 'Nuclear Issues',
				month: {
				$month: {
				$dateFromString: {
				dateString: '$Published Date'
				}
				}
				},
				year: {
				$year: {
				$dateFromString: {
				dateString: '$Published Date'
				}
				}
				}
				}}, {$match: {
				prediction_final: 'News'
				}}, {$limit: 5}, {$out: 'final results'}]	