Perspective API Project Report

Aishwarya Manojkumar & Caoilin Donnelly

Due: May 10, 2021

1 Background

1.1 Purpose of Perspective API

Perspective API is an ADS that utilizes machine learning models to flag abusive or "toxic" user comments online. This is done by being given an input of a comment (as a string of words), and the machine learning models assess the comment in eight different categories:

- **Toxicity**
- Severe Toxicity
- Insult
- Sexually Explicit
- Profanity
- Likely to Reject
- Threat
- Identity Attack

The analyzed comment/text is then rated in each of the eight categories via a score from $[0, 1]$, with zero being not representative at all, and one being that the comment fully embodies that categorization. Below is an example image that displays how the ADS works, from Perspective API's own website:

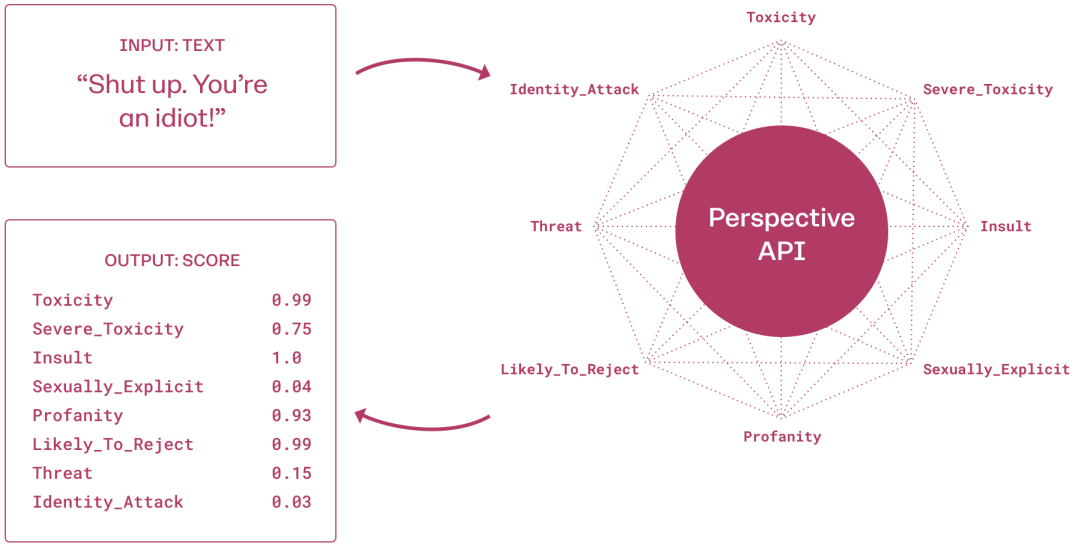


Figure 1: Visualization of Perspective API at work

The stated goals of Perspective API (from their website) are: To give developers and publishers the tools to give feedback to commenters, help moderators review comments easier, or help filter out "toxic" language. Perspective API defines "toxic" language as being: "rude, disrespectful, or unreasonable comments that are likely to make someone leave a discussion" (Perspective API).

It seems that the biggest implementation through their case studies & website is that of online moderation – where moderators can see the overall Toxicity score that perspective API calculated

on a given comment, and can then use that score to easily (and with greater speed) determine if that comment violates their rules, or is too "toxic" to be published on their platform.

1.2 Trade-offs of Perspective API

The biggest trade-off with perspective API is the inability to choose which of the eight categories (as a developer or moderator) you want perspective API to pick up on. An example of this would be an app such as Tinder may want to moderate unsafe or inappropriate profiles. Considering you must be 18+ to use Tinder, you wouldn't want something such as "profanity" to necessarily be flagged, considering everyone is an adult & should be able to handle profanity. However, the moderators or devs at Tinder won't have the option to filter on everything except profanity – and the profanity in a profile or message may still be flagged as "toxic," even if there is no inherent toxic intent.

This then begs the bigger trade-off of NLP in general, and how language is complex. Words carry different meanings, and although it is good to "analyze" specific details or words, context is important. And it seems that the ADS may or may not be able to easily interpret context (as a machine), and therefore a potential trade-off would be the limitation of NLP technology & classification.

2 Input & Output

2.1 Data Description

Perspective API, in general, is a free to use API software with a variety of applications (as seen above). However, this report will primarily be focusing on the specifics of the Kaggle competition we chose, and its input/output files and direction.

The data used by this ADS in the Kaggle competition was an example file of real, scraped data from Wikipedia's talk page edits (aka where the data came from). These comments are then placed into the "comment_text" column. There is then six different categories, representing six of the eight different categories Perspective API uses: toxic, severe toxic, obscene, threat, insult and identity hate. Each category is given a score of either 0 or 1, where 1 = the given comment is considered to be of that category,and 0 being not applicable. This is slightly different than that of perspective API's general usage, where a score is given on a scale between 0 and 1 per category. An example of what the dataframe looks like is below:

Table 1: First five entries in the dataset

	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	Explanation Why the edits made under my userna...	0	0	0	0	0	0
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	Hey man I'm really not trying to edit war. It'...	0	0	0	0	0	0
3	More I can't make any real suggestions on imp...	0	0	0	0	0	0
4	You sir are my hero. Any chance you remember w...	0	0	0	0	0	0

2.2 Input Feature Descriptions

This is mostly covered in the above section, and Table 1 shows how the data is laid out. In terms of data types, the comment_text is a string, and the other categories for each of the "toxicity" categorizations are integers. In the table above the first few comments just happen to not be toxic whatsoever, but as you will see in the pi chart below (done via the Kaggle competition code, not our code) that the majority of the almost 160,000 entries in the dataset are considered to be non-toxic (78%, to be exact):

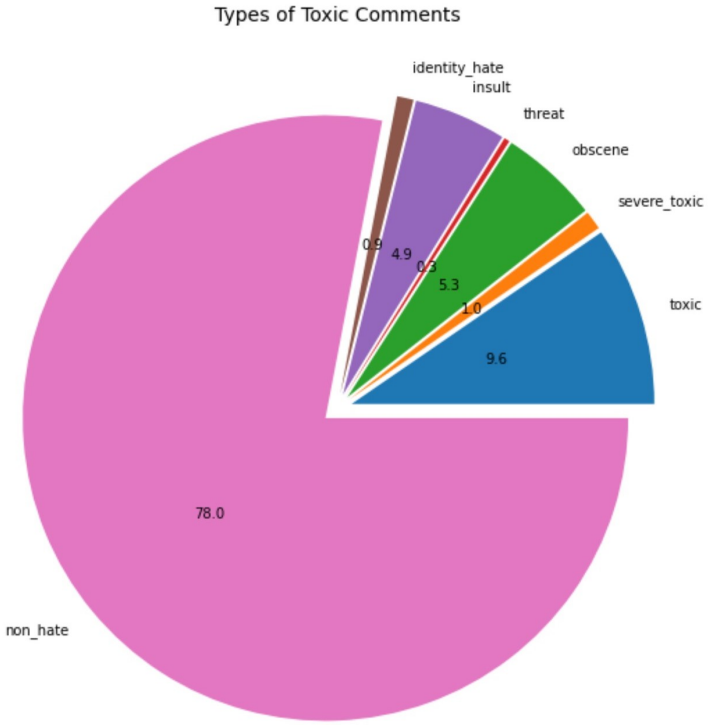
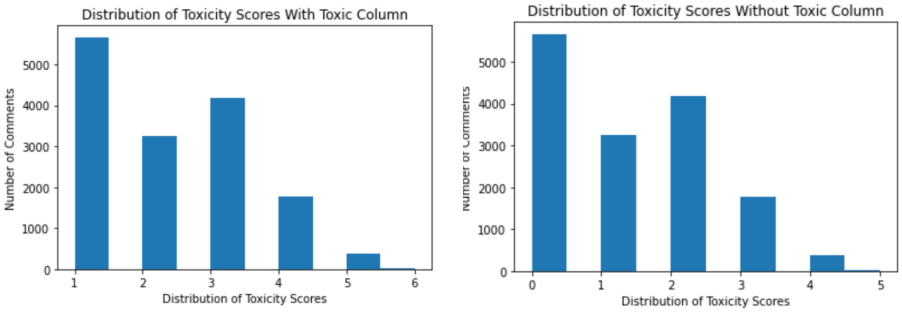


Figure 2: Pi Chart Breakdown of the Types of Toxic Comments

With that being said, there are instances where one or more of the toxic columns are present at the same time – as in, there doesn't need to just be "one" category, as a comment can be multiple different types of "toxic" at the same time. This is conveyed in the analysis we did below on a subset of the data where toxic = 1. The two diagrams show instances where toxic was included in the total score (e.g. the sum of all of the values in each of the six toxic categories), and when it was not:



As can be seen in the two figures, the most common flag for a comment to have is being toxic – which makes sense, looking at the pi chart distribution. But even more surprising is it is additionally quite likely for a comment to have up to 3 different categorizations at the same time. Toxic being the most popular classification can also be shown when it was not included in the sum, and the most popular category was no additional markers besides toxic – however the graphs are the same, just shifted over by a factor of one.

Lastly, we did one more analysis (which perhaps is maybe better-suited for section 4, but we thought was telling & interesting), and that is a cosine similarity between a subset of comments

with the same common categorization (in our case, we focused on the "threat" categorization). We chose threat because it was the smallest, and when we attempted to use larger ones such as "toxic," unfortunately the RAM necessary to do so crashed Google Colab, and therefore we used a smaller category.

A cosine similarity value of 1 would mean that the comments are identical, whereas a value of 0 would mean there is absolutely nothing in common. Stop words were also removed, so it only compared comments based on unique words (to ensure the cosine similarity is more accurate).

	0	1	2	3	4	5	6	7	8	9	10
count	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000
mean	0.049518	0.018716	0.034255	0.059444	0.073224	0.058026	0.024030	0.066783	0.056911	0.043516	0.061394
std	0.093874	0.063407	0.073879	0.096293	0.107375	0.111873	0.077544	0.091447	0.074252	0.081110	0.083747
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.026608	0.048507	0.000000	0.035143
75%	0.082516	0.000000	0.055272	0.103875	0.117720	0.089385	0.000000	0.115470	0.089443	0.077152	0.109109
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 3: Cosine Similarity Result Matrix (Sampling)

The results indicate that there is little to no similarity between all comments that were labeled as 'threat'. This is pretty interesting, as it then lends to the interpretation that what goes into flagging a comment as a 'threat' may not be the comment itself, considering how low the similarity is. This may mean that it is determined by 1-3 specific words (rather than overall or the context), or by another means. All in all, this analysis was effective in seeing how truly similar comments of the same category rating are categorized – which may inform us of the direction to go in as we continue with the project.

2.3 Output of the System

In terms of the ADS in general, as mentioned in the Background section, the output would be a value between 0 and 1 to represent the percentage of likelihood that a given comment or string is representative of that categorization or not. The output, therefore, is an overall "toxic" score as a number between 0 and 1, and then a breakdown of the other percentages per category.

In the Kaggle competition, this output is seen as a csv file with a few key attributes/columns:

1. The comment_text which is the actual comment being assessed (in this case, paired to a unique ID in the output file)
2. A score between 0 and 1 (inclusive) per each sub-population, as mentioned above

A comment can be within multiple sub-populations at once, as mentioned above. Below is a table showing the first 5 outputs of the system for reference of the above explanation:

Table 2: Example System Output

Comment ID	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity Hate
00001cee341fdb12	0.992	0.136	0.991	0.017	0.875	0.155
0000247867823ef7	0.009	0.003	0.006	0.001	0.009	0.003
00013b17ad220c46	0.036	0.004	0.014	0.002	0.017	0.005

3 Implementation & Validation

3.1 Data Cleaning & Pre-processing

The solution originally tried lemmatizing and stemming to clean the data, however they determined that the ROC-AUC score was too low when doing so. Lemmatizing and stemming are both ways to get to the root of words, rather than using the full words themselves. For instance, Walking and walked both have the same root (or "stem") of the word "Walk," so those words would be classified as walked. Stemming (such as a lancaster stemmer) is useful when a standard dictionary is used for the data at hand. Lemmatizing can be more accurate, but is best used with (1) a lot

of data, and (2) you are able to set up your own custom stem-classifications. This plays into the ROC-AUC score though, which essentially states how accurate or likely a classifier is able to correctly classify a 0 as a 0, or a 1 as a 1 – with that being said, one would want the highest ROC-AUC score as possible, so if lemmatizing/stemming yielded a low ROC-AUC score, it makes sense that this solution decided against such NLP techniques.

Instead of lemmatizing/stemming, they decided to clean out any standard "HTML" tags, such as: HTML line breaks, the commenter's Username, IP addresses, and HTTP links. They did so by replacing the following information with just removing the information & passing the rest of the comment string back into the data set.

3.2 High-Level System Implementation

After the data cleaning and pre-processing, the data is split into test and train sets for further classification and analysis. Upon being split into their respective sets, a tf-idf vectorizer is instantiated, fitted and transformed on the training set. Then, a logistic regression is run on each of the features separately. Next the training data is split into test and train data and is fitted to a GridSearchCV model. After that a vectorizer transforms the test set and predict_proba is called and the results are written to a new csv file containing the stated output in 2C.

3.3 ADS Validation

The solution code utilizes what is called a "classification_report" to get the precision, recall, f1-score and support for each sub-population (toxic, severe toxic, etc...) in the population. They also did a confusion matrix output for each sub-population as well.

Looking at the confusion matrix outputs, the classifier isn't perfect, but it does a pretty decent job at correctly identifying whether documents are classified correctly or not. An example of a confusion matrix for one of the sub-populations can be seen below:

Table 3: Confusion Matrix for the "Threat" Sub-population

31815	10
76	14
Accuracy:	0.998

As you can see, via the confusion matrix above for the "Threat" sub-population, there is an accuracy of 0.998, which is incredibly high for the classifier. Outside of just the confusion matrix, below are the precision results per each sub population via the "classification_report" analysis:

Table 4: Precision per sub-population

Sub-population	**Precision Score**
Toxic	0.95
Severe Toxic	0.99
Obscene	0.98
Threat	1.00
Insult	0.97
Identity Hate	0.99

Via these different tests seen in both tables 3 & 4 above, we can conclude that the ADS was thoroughly validated, and through the high precision and accuracy scores calculated we can conclude that it likely meets its stated goals.

4 Outcomes

4.1 ADS Accuracy Across Sub-populations

In part 3C the results of the classification_report and confusion matrices are reported. However, we did an additional accuracy test utilizing sklearn's accuracy_score function. We did this test on all six sub-populations, and the results can be seen below:

Table 5: Accuracy per sub-population

Sub-population	Accuracy Score
Toxic	0.95578
Severe Toxic	0.99113
Obscene	0.97782
Threat	0.99731
Insult	0.97055
Identity Hate	0.99210

Overall, we are pleased with the results of the accuracy_score calculations. The lowest score was for the "toxic" sub-population, in which the accuracy was still right around 0.96 (rounded up). The majority of sub-populations had an accuracy around 0.99, which is incredibly high. Because the accuracy was the lowest for the "toxic" sub-population, these results are the ones that guided our process in 4B when it came to fairness and diversity.

4.2 Fairness & Diversity Measures

Going off of our results from 4A, it was pretty apparent that all sub-populations are highly accurate. With that being said, there were slight differences amongst sub-populations – such as three of them (severe toxic, threat and identity hate) scoring above 0.99, and the other three (toxic, obscene and insult) scoring around the 0.96 - 0.97 range. Of course, neither are values to scoff at, but given how accurate the classifier already is, we nitpicked a bit at seeing if the classifier fairly assesses all sub-populations due to these minor differences.

The way we decided to observe this was through the confusion matrix established for each sub-population. This would show us if some sub-populations were more likely to be misclassified than others, considering the total number of classifications was about the same across majority of the sub-populations. The results can be seen in the table below:

Table 6: Confusion Matrix Results per sub-population

Sub-population	True Negative	True Positive	False Negative	False Positive
Toxic	28766	1753	161	1235
Severe Toxic	31558	69	54	234
Obscene	30209	1026	82	598
Threat	31815	14	10	76
Insult	30235	748	177	755
Identity Hate	31617	44	21	233

As can be seen above, there are a lot of interesting results. Below is a list correctly representing what each attribute represents, pertaining to our data:

- **True Negative:** Comments that were correctly classified as not a part of that sub-population (e.g. "not toxic")
- **True Positive:** Comments that were correctly classified as part of the given sub-population (e.g. a "toxic comment")
- **False Negative:** Was labeled as a non-toxic comment, when in actuality it is a comment that exhibits toxicity & should be reviewed
- **False Positive:** Was labeled as a toxic comment, when in actuality it is a comment that is non-toxic & doesn't require review

Despite the classifier having a very high accuracy, it can be seen that some sub-populations have significantly higher false positive & negative rates than other sub-populations. For example, "toxic" has the lowest accuracy, and it also has the highest false positive rate (and second highest false negative rate). This is also true of the other two sub-populations in question, insult and obscene, which also have the highest false negative & false positive rates. All three of the sub-populations we had questions about, unsurprisingly, had the top three highest false negative & false positive rates.

With that being said, our hypothesis behind why that is the case is not a matter of "fairness," but population size & how "broad" the sub-population is. While all three had the top three FN and FP rates, they also had the highest number of true positives – aka the highest number of instances

of a comment meeting that sub-population's criteria. By name, they also seem to be the three most broad topics (which makes sense given their true positive rates), which may be why their accuracy is lower.

An example of this is looking at the identity hate column. Identity hate is described as being harmful against one's identity, such as homophobia, racism, sexism, etc... So, as one would assume, to be classified as "identity hate" one would have to comment something using some pretty specific words or slurs towards these minority identities or communities. This may be why it is such an accurate metric, in comparison to the other five. This can also be said of "threat" and "severe toxic" as well, as they are more specific.

However, something like obscenity or toxicity or just an insult are likely harder to define in the NLP scope. More words contribute to something being "potentially toxic," and therefore it may inherently be harder to correctly classify those sub-populations. Due to the accuracy of the model, and that it is a pretty small difference (relatively speaking) between sub-populations, we believe that the classifier/algorithm is indeed fair; And any discrepancies are because some categories are inherently broader than others, and therefore become harder to pinpoint & classify.

4.3 Additional Methods for ADS Performance

The additional method we decided to look at was further dissecting the classification_report statistic, as well as the actual ROC-AUC score. Looking back to earlier sections, the classification_report shows the precision, recall and F1 score for each of the sub-populations individually. Below is a figure showing the full classification_report for all sub-populations:

Toxic Classification report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	28927
1	0.91	0.58	0.71	2988
avg / total	0.95	0.96	0.95	31915

Severe Toxic:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31612
1	0.58	0.24	0.34	303
avg / total	0.99	0.99	0.99	31915

Obscene:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	30291
1	0.92	0.62	0.74	1624
avg / total	0.98	0.98	0.98	31915

Threat:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31825
1	0.58	0.17	0.26	90
avg / total	1.00	1.00	1.00	31915

Insult:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	30412
1	0.81	0.49	0.61	1503
avg / total	0.97	0.97	0.97	31915

Identity Hate:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31638
1	0.68	0.17	0.27	277
avg / total	0.99	0.99	0.99	31915

Figure 4: Classification_Report Results for all Sub-Populations

Recall is defined as the ratio of correctly predicted positive observations to all observations in an actual class. The recall values for all six sub-populations are between $[0.96, 1]$, which is great for

this model. **Precision** is defined as the ratio of correctly predicted positive observations to the total predicted positive observations (as was further discussed in 3C). Lastly is the **F1 Score**, which is the weighted average of precision and recall. This score then takes both false positives and negatives into account – the highest f1 score possible is 1 (perfect recall and precision), and the lowest is 0 (if either recall or precision are zero). It is essentially the composite of both scores.

Lastly, we did an additional test to get the true ROC-AUC value, which is how likely the model is to correctly classify any given document (or comment, in this case). To refresh, a high ROC-AUC score means the model is very accurate (closest to 1 as possible), whereas a low one would indicate the model, as a whole, does not perform as intended. The ROC-AUC score we calculated came out to **0.865**, which is pretty decent, but a bit lower than we expected (considering how accurate the sub-populations were). But this is an important statistic because it evaluates the performance for the ADS overall, and not necessarily based on features.

5 Summary

5.1 Appropriate Data for the ADS

The data used definitely seems appropriate for the ADS we chose. As was seen in parts 2C and 1A, the Perspective API output and inputs seem to be exactly the same as the data that was used during the Kaggle competition. Perspective API takes in a comment, and as mentioned gives a score from 0 to 1 (inclusive) per sub-population, but also an overall "toxicity" score.

With that being said, there are two small differences between the Perspective API output & that of the Kaggle competition. The first is the lack of the Kaggle competition having an overall toxicity score, and the second is the Kaggle competition not including three categories that Perspective API implements (profanity, likely to reject, and sexually explicit), and instead includes the category "obscene."

The data still seems to be very appropriate for the ADS at hand, however small changes could be made to be fully in-line with the Perspective API algorithm.

5.2 Implementation Assessment

The implementation certainly seems to be robust, accurate and fair. As can be seen in earlier parts, we (along with the solution) used a variety of different accuracy and fairness measures to assess the models. Such methods were: sklearn's accuracy_score method, classification_report (for the precision, recall, etc...), and multiple confusion matrices per sub-population. All of these metrics returned extremely high values for every single sub-population, with the lowest being a value of 0.96 – which is still incredibly high and accurate for the classifier's performance. From that, we can conclude that the implementation certainly seems to be robust, accurate and fair.

In terms of stakeholders, likely the vendor, data holder, and the moderators (or users of the API) would benefit the most from these results. The data holder benefits because their system is robust and accurate, and therefore more valuable to potential clients/vendors (bringing in more company revenue). For the vendors, having an accurate algorithm significantly benefits their team (which includes the moderators/users). Because the implementation is meant to help moderators rather than be a moderation replacement, accuracy helps moderators do their job quicker and correctly, with less feedback and complaints for the forum users who may have their comments removed. If the moderators are more effective at their jobs, then the company also wins because they can get more done for the same amount of pay, and also get less user backlash because their moderation decisions are quite accurate.

5.3 Public Sector Implementation

Perspective API is already used in industry, with some of their clients including *The New York Times (NYT)*, the *Southern Missourian* (and other local newspapers), and *Coral by Vox Media*, which includes tools for all major newspapers (The *Washington Post, Wired, Wall Street Journal (WSJ), USA Today, LA Times* and more. Considering it is already deployed in the public sector, we would say that is is okay for the industry. It already has improved moderation capabilities by 60% at companies like the NYT, coupled with the accuracy& fairness measures we calculated, it definitely seems suitable for the public sector in industry.

With that being said, it would be in Perspective API's best interest to continue to improve the API (thus the Kaggle competition, to hopefully improve the API). One potential improvement users

have expressed is being able to turn off certain sub-populations, because on websites with older audiences they don't want something like "profanity" to classify an otherwise okay comment to be potentially removed. But even without that, the API has proved to be successful in the public sector, and with any algorithm hopefully they are able to update it over time to further improve it. But we do support its use in industry, considering it is highly accurate per the metrics we predicted.

Lastly, the biggest thing to look out for would be emergent bias. Perspective API is meant to be a tool used in conjunction with human moderators, and their scores taken into account when assessing whether to remove hateful comments or not. However, it is important that the human moderators still use their own judgement when moderating, and don't get to a point where they rely too heavily on the scores from Perspective API to make those important moderating decisions.

5.4 Improvements

Potential improvements were already partially provided in other sections of this report, so the below is more of a recap rather than new suggestions.

For starters, I think it would have been good to accurately reflect all categories given by Perspective API. This would include adding assessments in the processing phase to cover profanity, likely to reject, and sexually explicit – as well as removing "obscene." Doing so would better reflect the actual API, which would strengthen the applications of the implementation.

Additionally, the implementation does not feature an overall toxicity score, which is an important metric in Perspective API. This is the value that moderators primarily look to (rather than the individual sub-population scores), and the implementation not having that does make it less usable in terms of deploying this specific solution as a potential upgrade to Perspective API. So, it would be good to add this metric back in – again, to strengthen the application of the implementation.

Lastly, as mentioned briefly in part 5C above, the actual ADS that is Perspective API could use a filtering feature when assessing sub-populations. Different vendors should be able to choose which categories they want to feed into the overall toxicity score, or which ones they want to search for when moderating. Not all websites moderate or function the same, so giving vendors that option would be a step in the right direction for improving the ADS to be more customizable, and work for a wider range of situations.