AP4

1. Data splits:

We chose random splits, as we already took care when creating the dataset to diversify it. The number of posts per user and sub were limited, so a random sampling should give us good coverage in every set. I checked the distribution of labels and each split, and they seemed roughly equivalent to the distribution in our adjudicated set.

I used pandas data frames to read the adjudicated TSV, and then sampled with pandas, and split with numpy. Then when writing out the splits, we had to omit the adjudicator label, as for training we only care about the doc_id, text, and gold labels. I did this within the pandas data frame, and then used the data frame to write each TSV.

2. Building predictive models

Baseline:

A majority class classifier would have an accuracy of 0.497, since the majority class in the training set is “5” and the test set has 103 documents with label “5”.
The ordinal regression model with BOW features has an accuracy of 0.502 with a confusion matrix:

		prediction				
		1	2	3	4	5
gold label	1	0	0	1	1	12
	2	0	0	0	3	16
	3	0	0	4	1	24
	4	0	1	4	0	40
	5	0	0	0	3	103

Proposed model:

We tried three approaches.

Baseline Bert (not that great):

Our first stab at a baseline BERT training didn’t go too well. We got an accuracy of 0.500 which barely edged out choose most common, and performed the same as baseline linear regression. This was surprising at first, especially when we got an accuracy of 53.2% from layer 4 of the unsupervised pretrained BERT model. It did, however, perform better than baseline ordinal linear regression, especially with 1 labels, of which BERT caught some and OLR got none. This, combined with better performance with a custom tokenizer and df set to 3, lead us to believe

that if we added custom features to OLR, we may see further improvements. As we'll discuss in the next section, this didn't pan out as well as we'd hope.

Ordinal regression with refined features:

We trained an ordinal regression model on 2 primary features: bigrams+trigrams, and bad words. Our idea with using ordinal regression was to be able to train the classifier on the dataset using the natural order of our labels: 1-5, where 1 means not family-friendly while 5 means very family-friendly.

To aid featurization, we preprocessed docs into tokens—we split sentences, normalized cases for each word, lemmatized each word and removed stop-words.

Final OLR Results:

We used both these features together to create train, dev and test vectors composed of a Sparse Matrix which could be fed to the classifier.

The features together reached a maximum accuracy of 53.5%, with a CI of [0.498, 0.604]. The reason for the low accuracy has to do with the size of the corpus and the labelling itself. We found that all the predictions were 5, meaning that the data was inadequate for OLR and the labels were not very helpful for the features we selected.

In hindsight, if we had thought of these features (and classification as a whole) while developing the annotation guidelines, our classification could have performed better, since we would have focused on labelling things like incendiary words with a lot more weight than trying to consider the overall context of each doc.

Fine-tuned Bert:

We figured using RoBERTA could improve BERT model performance, as it was trained on more data, and we also hoped that Facebook also trained it with social media comments and texts, which are more relevant to our task.

We used our dev_set to tune the hyperparameters, initially starting 4 seeds and learning rates between 1e-6 and 1e-3, then gradually narrowing this window to find the best hyperparameters for both.

We trained both BERT and RoBERTa with a resulting test_accuracy on the test_set with 0.549 - 95% CI [0.482, 0.616] and 0.559 - 95% CI [0.492, 0.625] accuracy respectively. Although we believed RoBERTa might be superior due to this accuracy initially, upon closer inspection, BERT had better recall and precision in general, especially for the important not family-friendly posts with a label "1".

BERT model, confusion matrix, precision, and recall:

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Recall
Gold 1:	7	1	2	2	2	0.500
Gold 2:	5	1	6	5	2	0.053
Gold 3:	2	0	7	8	12	0.241
Gold 4:	1	0	5	15	24	0.333
Gold 5:	0	0	6	13	87	0.821
Precision	0.467	0.5	0.269	0.348	0.685	

RoBERTa model, confusion matrix, precision, and recall:

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Recall
Gold 1:	0	2	9	1	2	0
Gold 2:	0	4	9	1	5	0.210
Gold 3:	0	0	13	2	14	0.448
Gold 4:	0	0	14	8	23	0.177
Gold 5:	0	0	9	3	94	0.886
Precision	0	0.666	0.241	0.533	0.681	

What we can see from here is that documents with labels “5”, “4”, “3” are hard to distinguish apart, suggesting that mild insults/rudeness (common reason for documents to have a label of “4” or “3”) is not an easy feature to use.

3. Analysis

On Features:

OLR Features

Bigrams and Trigrams

To enforce context, we chose to use n-grams with 1 < n <= 3. We experimented with different combinations of unigram, unigram+bigram, bigram, bigram+trigram, trigram and found that the bigram+trigram feature set performed better than the others with an accuracy of ~53% itself.

The combination of all 3 together had the lowest accuracy, followed by unigrams alone (bag of words), which could not meet the baseline BeRT’s baseline of 53%.

Incendiary Words

Since our task was focused on discriminating between a post/comment that would be family friendly and one that is not, we could leverage simple incendiary words to create a feature set of bad word presence.

This featurizer consisted of a static list of words we considered incendiary, that would almost always point towards a post/comment being not family-friendly. These words were drawn from our annotation guidelines. Every time a token was seen in a document matching any word in this list, the features set marked it as present (1 or 0).

This feature would determine if certain incendiary words were present in a doc, and if so, what combinations of those.

The results here were also undermining with an accuracy of ~51%, varying depending on what words were added to the list. However, this may not mean that the featureizer is weak, but that it did not work well on the current corpus and labels.

LR Feature weights

We were able to easily extract the learned features from regular LR with binary bag of words, but a df of 3 and an improved tokenizer. The learned features for 5 scores are unsurprisingly very general. This is to be expected because our annotation were purely subtractive. This means that the only real defining feature of a 5 score, is that it wasn't scored a 4 or lower. None of the learned terms actually contributed to a 5, but an absence of other words did. The list for 5's is as follows:

5 many 0.418
5 sure 0.326
5 made 0.320
5 way 0.265
5 much 0.243
5 post 0.242
5 best 0.236
5 course 0.227
5 work 0.220
5 idea 0.218

In the 4 category, we start to see some of our annotation guidelines being recognized. Terms like 'hell' and 'shit' that are often used in only mildly offensive ways crop up here. This is likely because we were very context conscious with profanity. If the words were used sparingly, in a way that wasn't targeted at any individual, and contained no other guideline violations, regularly received a score of 4. Thus, profanity terms themselves weren't a very discriminating feature. The top weights for 4 are as follows.

4 hell 0.320
4 since 0.312
4 said 0.303
4 sense 0.288
4 nothing 0.284
4 world 0.274
4 world 0.274
4 seen 0.272
4 shit 0.269
4 team 0.268
4 get 0.256

Scores of 3 were often categorized as such because they were unnecessarily rude or mean, but not containing any specific terms that were deemed ‘bad’ when we were annotating. It’s unsurprising that this set is mostly like 5, and doesn’t contain terms that are particularly meaningful, as it was mostly context that went into documents with 3 scores. The list is as follows:

3 u 0.359
3 seems 0.341
3 people 0.301
3 away 0.298
3 give 0.290
3 person 0.280
3 power 0.237
3 think 0.221
3 least 0.217
3 child 0.214

Scores of 2 were much less common than the previous 3 scores, and they frequently made category 2 in our annotation guidelines. However, category 2 also contained lots of contextual information, and we typically didn’t subtract mentions of specific words to a 2 label if the context was not glorifying or encouraging these things. Things like mentioning alcohol or cannabis was considered fine by our standard (score of 3 or 4), but the excessive use or encouragement of them was considered bad. This is further evidence that BOW models have a really hard time dealing with context-specific things, something our guidelines relied heavily on. The top weights for 2 are as follows:

2 fuck 0.367
2 without 0.337
2 man 0.246
2 day 0.235
2 country 0.232
2 friend 0.230

2 little 0.228
2 take 0.213
2 dude 0.211
2 men 0.210

Finally, scores of 1 were considered ‘category 1 terms’ by us in our annotation guidelines. This means that subjects that mentioned these things, except for extenuating education circumstances without gratuitous detail, would always score a 1. Therefore, this is one of the most cohesive categories of words. Oddly enough, the model that learned these weights failed to classify anything as a category 1. This is likely due to limitations in training data. Because of Reddit moderation, there weren’t many posts with these labels, and so, learned terms were likely heavily biased towards the training set, and failed to find documents with these terms in the training set. The weights for 1 are as follows:

1 blood 0.307
1 maybe 0.267
1 sex 0.234
1 suicide 0.222
1 fact 0.221
1 someone 0.199
1 fuck 0.182
1 understand 0.174
1 wrong 0.169
1 mean 0.167

On Mistakes:

In regards to BERT:

Our BERT model notably mislabeled 3 labels more than other labels. On further analysis, we found that 3 labels were often mistaken for 2 and 5 labels. One notable example of a mislabel was a document that contained the quote “Talk about traumatic…Got stung hundreds of times, cause the don't die after one sting, those fuckers keep going.” While our gold label for this document was a 4, since it contains the word “fuckers” and loses it one point, BERT predicted that this document was a 1. This is likely due to the fact that it took into account the words “fuckers,” “die,” and “traumatic,” all of which are generally negative words. In this document, “die” and “traumatic” are used in a descriptive and factual way, and should not necessarily reduce the document’s label, but are generally label-reducing words.

Another example of a mislabeled document was the document containing the quote “Wow…So you don't bear in mind that the person who owns the car has tremendous personal and financial responsibilities? Where else are you going to get a free ride in a personal car? $20 is CHEAP for a 35-45 min ride when you compare an Uber being $50-$75.” This document had a gold label of 3, but was predicted to be a 5 by BERT. This is very likely due to the fact that this document is written with a rude and demeaning tone, which was difficult for the model to understand. The writer is clearly implying that the person he is replying to is being ungrateful and taking advantage of the person they are getting a ride from, but it uses no words that are necessarily label-reducing like explicit language or level-1 words. It was a fairly common

occurrence for the model to mislabel documents that were rude in an indirect or condescending way if they didn't use words that were necessarily bad, aggressive, or explicit.

All in all, we think there are two major types of mistakes we can fix – the first one is that context from previous posts users are replying to are completely missing, and the second being that the database might not be big enough. Other observations we made that may be a factor, or might be hard to fix are:

1. Speaking rudely in a demeaning tone does not necessarily have to use "bad" words
2. People on social media put a lot of words in quotes to express a different meaning
3. Educational posts that use "bad" words, which can be very hard to classify from a computer's perspective.

On Model Biases:

Unfortunately, we didn't save any metadata or even Reddit's internal post_id to recover metadata from our documents. In hindsight, this would have been very useful because we could have compared accuracies (or more importantly for our project, recall in the low scoring documents) across different subs. We also could have looked at the scores of these comments, to see if upvote count was related to family friendliness, or if they were related on a 'sub'population level. It would have also probably been useful to see if the model (or us humans annotating without the metadata) were getting confused by things like cartoon violence on gaming subs.