Cross-Domain Detection of Hate-Speech/Toxicity

Noam Elul and **Aman Kumar** and **Dan Gallagher** and **Yuling Liu**
University of Pennsylvania

Abstract

In this project, we evaluated the cross-dataset performance of hate-speech classifiers. We trained classifiers on labeled data from one website, and evaluated their performance on data from another website. If we can build a model that succesfully transfers from one dataset to another, we may be able to use those same models on websites where we have little to no labeled data. We find that there is some potential promise in the ability for BERT-based hate-speech detection models to transfer across contexts and even text structures.

1 Introduction

Detecting hate-speech and toxicity has been a topic of increasing relevance, particularly for many social media sites. The ability to classify comments as different levels/types of toxicity helps content moderators and filters more accurately and efficiently decide which comments to remove.

While there exist many labeled datasets classifying toxic or problematic comments, it would be difficult to generate those datasets for every different website/forum. Our goal was thus to train toxicity classifiers using datasets from some websites, and measure their performance on datasets from another website. The idea being that if a classifier is able to generalize from one website where we have data to another, it might also generalize well to websites where we don't have data. This is particularly relevant in a modern world where new social media sites are constantly arising, with some fulfilling niches that may be particularly vulnerable to hate-speech. In these contexts, it would take time and effort to properly annotate data, so being able to use a model trained from a different social network in the interim could be highly valuable in preventing the spread of hate-speech.

The table below illustrates our example input and output for the task.The binary classifier is given text from multiple websites and determines if the text is hate-speech (True) or not (False).

Input Text	Output
Don't judge me.. i am vindictive and vengeful when it comes to bitches disrespecting me	True
It should definitely say something about Kyle Vander Wielen and all his bitchin	True

2 Literature Review

2.1 Vidgen, B., Nguyen, D., Margetts, H., Rossini, P., & Tromble, R. (2021)

In order to address the limitations of previous labeled abuse datasets and improve the quality of annotations, Vidgen et al. (2021) created a new annotated dataset of 25,000 Reddit text data. The dataset contains a hierarchical taxonomy of abusive content with 3 primary categories for abusive content (identity-directed abuse, affiliation-directed abuse, person-directed abuse), 3 primary categories for non-abusive content(neutral, counter speech, non-hateful slurs), and additional secondary categories.

Vidgen et al. (2021) used community-based sampling to get a more realistic dataset. Entries from Reddit post titles, bodies, and comments are annotated and are assigned with at least 1 category by two human annotators. Vidgen et al. (2021) also considered every entry's context- an entry that appears to be non-abusive by itself but is abusive when context is considered can be labeled correctly. However, the classification is highly skewed toward neutral (there are only a few non-hateful slurs).

Vidgen et al. (2021) separated 23417 entries into 58% training, 19.3% development, and 22.7% and experimented the data with several baseline models, including Logistic Regression, BERT. A high F1

score was achieved on Neutral entries possibly due to skewed data. Context can be taken into account for prediction as future work.

2.2 Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April)

Uses data from Waseem and Hovy (2016) – roughly 16,000 tweets that were labeled as racist, sexist, or neither (though seemingly nothing labeled as both racist and sexist). This was seemingly an initial exploration of using deep learning to detect hatespeech. Due to its time of publication, it did not make use of transformer architectures.

Deep learning methods outperformed "traditional" machine learning classifiers used on different representations of the text, including character-based n-grams, bag-of-words, and tf-idf. They tested GloVe embeddings as well as random embeddings, and interestingly found that random embeddings tended to work better.

Perhaps the most key point: performance significantly improved when using the average of word embeddings learned by the neural network as features for a gradient-boosted decision tree.

2.3 Schmidt, A., & Wiegand, M. (2017, April)

This paper is a survey on methods and features used for automatic detection of hate speech. It discusses the pros and cons of each of the methods/features and if there is room for further research in those areas. The initial analysis is of using generic features like bag of words and embeddings which lead to decent classification performance. Character level approach yields better results than word/token level one. Sentiment information can be used as a feature where negative sentiment would correlate to hate speech.

Lexical features like a list of slurs or insults can improve classifier performance. Some linguistic feature methods like dependency parsing are effective and others like POS tags are not so much. Incorporating multimodal features and meta data alongside the text data has led to better results on social media platform based hate speech analysis. For modeling, in Machine Learning SVM classifiers and in deep learning RNN based language models produce the best results.

3 Experimental Design

3.1 Data

We utilized hate speech data from four different sources to assess how our models transferred across domains. Two of these datasets were comprised of tweets, thus exhibiting the same "structure," but had different contexts (one being general tweets across contexts and the other being specific to the 2020 US presidential election), while the other two domains were comprised of Reddit and Wikipedia comments, representing data of both differing context and structure. All the data was binarized into hate-speech and non-hate-speech classes in order to create a consistent annotation scheme for transfer analysis.

3.1.1 General Tweets

Davidson et al. (2017) compiled a random sample of tweets containing words and phrases identified by internet users as hate speech, compiled by Hatebase.org. These tweets are not specific to a certain context, and can thus be thought of as a general representation of hate speech on Twitter. Furthermore, Davidson et al. distinguished between offensive speech and hate speech, whereas many similar datasets lump the two together. Given that it was unclear what the "standard" for labeling text as hate speech should be, we constructed two separate datasets from this source. The "Loose" dataset includes offensive language as hate-speech, essentially offering a lower bar for what is considered hateful whereas the "Strict" dataset sets a higher bar and does not consider such language as hate-speech.

3.1.2 Political Tweets

Grimminger & Klinger (2020) fetched tweets for 6 weeks leading to the presidential election, on election data, and 1 week after. They describe their filtering process as using "the mention of the presidential and vice presidential candidates and the outsider West; the mention of hashtags that show a voter's alignment such as the campaign slogans of the candidate websites, and further nicknames of the candidates." Notably, the annotation instructions defined hatefulness in a way that likely lines up with the previously defined "Loose" bar, which includes offensive language some might not actually consider "true hate-speech."

3.1.3 Reddit Comments

Vidgen et al. (2021) introduced a new dataset of primarily English Reddit entries which addresses several limitations of prior work. It (1) contains six conceptually distinct primary categories as well as secondary categories, (2) has labels annotated in the context of the conversation thread, (3) contains rationales and (4) uses an expert-driven group-adjudication process for high quality annotations. Among the six primary lables, we mapped "Neutral" to non-hate comment/speech, and the others to hate comment/speech.

3.1.4 Wikipedia Comments

This dataset is taken from Kaggle toxic comment classification challenge, which was a part of a research initiative started by Jigsaw and Google to build tools that will help improve online conversation. The dataset contains comments from wikipedia's talk page edits, and has been labeled by human raters for different type of toxic behavior. The different type of toxicity labels in this dataset: toxic, severe toxic, obscene, threat, insult and identity hate. For the purpose of our project, we condense/preprocess these labels as a single label i.e. hate comment/speech.

3.1.5 Data Statistics

Though the general Twitter and Reddit datasets are of similar sizes, the political Twitter dataset is quite a bit smaller and the Wikipedia dataset quite a bit larger. Additionally, all but the Twitter (Loose) dataset were highly skewed, with far more non-hate-speech examples than hate-speech examples. Twitter (Loose) was also quite skewed, but in the opposite direction, possibly due to the lower bar in determining if a Tweet is hate-speech coupled with the fact that Tweets were pre-selected to have potentially offensive terms.

Dataset Sizes				
Split	Twitter (Both)	Political	Reddit	Wikipedia
Train	12391	2040	13585	127656
Val	6196	360	4527	31915
Test	6196	600	5308	63978

Table 1: The number of examples (tweets/comments) in each split of each dataset

Class Counts			
Dataset	# Hate-speech	# Non-hate-speech	% Hate-speech
Twitter (Loose)	10300	2091	83.12
Twitter (Strict)	743	11648	6.00
Political	243	1797	11.91
Reddit	2511	11073	18.48
Wikipedia	12980	114676	10.17

Table 2: The distribution of positive and negative examples for each dataset. We see that every dataset is quite imbalanced, though Twitter Loose in the opposite direction from the rest.

Data Examples		
Dataset	Hate-speech Example	Non-hate-speech Example
Twitter (General)	RT @Annnna_dav69: I hate trailer trash	The irony is so deep you can use a shovel.
Political	@greenfield64 No Way! @realDonaldTrump F** K Trump Wish him the best but you know what, he knew the risks and should have worn a mask. #BidenHarris2020 #screwtrump...	@SukiRavan @ProgressPotato @MarkZuckerb0rg @JSiffordson @KyleKulinski If Joe Biden is .000000000001 % better I will vote for him.
Reddit	Ol' Mikey was gay as all fuck. Leo too I think?	In 1971, it gave birth to Bangladesh; now Balochistan is on its way!
Wikipedia	You people are posting the WRONG results so its MY problem jackass	Fine. (Are we going to change all the others, too? No? Just asking.) Schmuckola could have saved a lot of grief by simply citing the policy, rather than strutting around like some self-important little peacock. (But that's Wikipedia; it has a tendency to attract such people. Whatever.)

Table 3: Positive and negative examples from each dataset

3.2 Evaluation Metric

We evaluated the performance of our models mainly using the f1-score, the harmonic mean of precision and sensitivity (recall for documents being classified as problematic). We also individually looked at precision, sensitivity, and specificity (recall for documents being classified as not problematic) individually to gain more insights. Specifically, precision for a class can be defined as

$$\frac{TruePositives}{TruePositives + FalsePositives}$$

and recall for a class as

$$\frac{TruePositives}{TruePositives + FalseNegatives}$$

. We used these metrics because they give a better sense of how our model handles class imbalance like that which we observed, where a model that completely ignores the minority class will still achieve high accuracy.

In our evaluation, we also included accuracy

$$\frac{TruePositives + FalsePositives}{\#Examples}$$

as a metric to get a sense of which ways a model might be succeeding, but it is important to note that a high accuracy in the absence of high performance on the other metrics does not signify a successful model .

3.3 Simple Baselines

We implemented three simple baselines:

1. We implemented a proportional classifier which classifies all instances as problematic with a probability equal to the percentage of training data samples that were tagged as problematic.

Proportional Classifier					
Dataset	f1	precision	sensitivity	specificity	accuracy
Twitter (loose)	0.836	0.834	0.838	0.178	0.730
Twitter (strict)	0.041	0.042	0.040	0.940	0.883
Political	0.186	0.194	0.179	0.891	0.794
Wikipedia	0.102	0.101	0.102	0.898	0.815
Reddit	0.176	0.179	0.173	0.812	0.698

2. We implemented a majority classifier, which classifies all instances as the majority class.

Majority Classifier					
Dataset	f1	precision	sensitivity	specificity	accuracy
Twitter (loose)	0.909	0.833	1.000	0.000	0.833
Twitter (strict)	0.000	NaN	0.000	1.000	0.941
Political	0.000	NaN	0.000	1.000	0.892
Wikipedia	0.000	NaN	0.000	1.000	0.898
Reddit	0.000	NaN	0.000	1.000	0.816

3. We found that the majority classifier was difficult to use since its performance

varies so wildly across datasets (and because it often gives undefined values due to predicting no positives), so we implemented an additional baseline, a classifier that classifies all samples as positive.

All-Positive Classifier					
Dataset	f1	precision	sensitivity	specificity	accuracy
Twitter (loose)	0.909	0.833	1.000	0.000	0.833
Twitter (strict)	0.098	0.051	1.000	0.000	0.051
Political	0.179	0.098	1.000	0.000	0.098
Wikipedia	0.183	0.101	1.000	0.000	0.101
Reddit	0.308	0.182	1.000	0.000	0.182

The all-positive baseline was useful since it gave the highest F1 score of the simple baselines, which made it a good choice for evaluating performance.

The proportional baseline was also useful since it gave nonzero values for all of our metrics, so it could be used for evaluating things other than F1 score.

4 Experimental Results

4.1 Strong Baselines

4.1.1 Strong Baseline 1: Naive Bayes on TF-IDF Representation

Our first strong baseline used a Multinomial Naive Bayes classifier on the Term Frequency/Inverse Document Frequency (TF-IDF) representation of the data. The TF-IDF representation is made up of two parts:

$$TermFrequency = \frac{\#\ of\ instances\ of\ word\ w\ in\ document\ d}{\#\ of\ words\ in\ document\ d}$$

$$InverseDocumentFrequency = log(\frac{\#\ of\ documents\ in\ corpus}{\#\ of\ documents\ containing\ word\ w})$$

The point of tf-idf is both to get a sense of how important a word is in a given document (based on how frequently it appears) as well as how much information a word might provide to a given document (based on the idea that, if a word is rarer across all documents in the corpus, it might be specifically important to a document where it is actually used). The final representation is then defined as $TF\ IDF = Term\ Frequency * Inverse\ Document\ Frequency$.

Based on the tf-idf representations of each word, we then use a Multinomial Naive Bayes (MNB) classifier to classify each document (tweet, comment, etc.) as problematic or not. MNB uses Bayes rule $P(A|B) = P(A)P(B|A)P(B)$ to calculate the probability of each class, based on the tf-idf representations of each word in the sequence. Importantly, Naive Bayes assumes the locations of the words in our documents are independent of one another, which is of course untrue, and one of the reasons why other methods might work better.

Multinomial Naive Bayes					
Metrics	Evaluation Dataset				
	Twitter Loose	Twitter Strict	Political	Reddit	Wikipedia
F1	0.916	0	0	0.004	0.023
Precision	0.846	nan	nan	0.5	0.1
Sensitivity	0.999	0	0	0.002	0.013
Specificity	0.099	1	1	0.999	0.986
Accuracy	0.848	0.948	0.901	0.818	0.888

Figure 1: Multinomial Naive Bayes Classifier trained and evaluated on all datasets.

4.1.2 Strong Baseline 2: LSTM

The second strong baseline is a LSTM based classifier. The data preprocessing step removes stopwords and punctuations and converts tokens to lowercase. Spacy tokenizer is used for obtaining tokens. The train dataset is used to create a vocabulary to id mappings. The text data in the train and dev set is encoded. Each sequence is then padded or truncated based on the desired sequence length.

$$o_t, h_t = LSTM_t(x_t, h_{t-1})$$

The default LSTM network architecture is a bidirectional LSTM with a fully connected layer. The selected embedding size is 300 and the learning rate for training models is 0.001, with batch size of 32 as these hyperparameters had the best results. The evaluation pipeline is used to evaluate the model on test sets from same/different domains.

Bidirectional LSTM (lr = 1e-3, batch size = 32)						
Training Dataset	Metrics	Evaluation Dataset				
		Twitter (loose)	Twitter (strict)	Political	Reddit	Wikipedia
Twitter (loose)	F1	0.966	0.103	0.2	0.314	0.154
	Precision	0.969	0.054	0.131	0.234	0.101
	Sensitivity	0.964	0.881	0.423	0.477	0.329
	Specificity	0.847	0.174	0.695	0.654	0.671
	Accuracy	0.944	0.211	0.668	0.622	0.637
Twitter (strict)	F1	0.046	0.249	0.057	0.094	0.057
	Precision	0.953	0.437	0.2	0.428	0.092
	Sensitivity	0.024	0.175	0.033	0.052	0.041
	Specificity	0.994	0.987	0.985	0.984	0.954
	Accuracy	0.186	0.945	0.891	0.814	0.862
Political	F1	0.221	0.092	0.269	0.132	0.096
	Precision	0.947	0.067	0.253	0.326	0.103
	Sensitivity	0.125	0.144	0.288	0.082	0.091
	Specificity	0.965	0.891	0.907	0.961	0.911
	Accuracy	0.266	0.853	0.846	0.802	0.829
Reddit	F1	0.127	0.04	0	0.006	0.008
	Precision	0.938	0.037	NaN	0.375	0.101
	Sensitivity	0.068	0.043	0	0.003	0.004
	Specificity	0.977	0.938	1	0.998	0.995
	Accuracy	0.22	0.892	0.901	0.817	0.895
Wikipedia	F1	0.853	0.107	0.336	0.341	0.129
	Precision	0.944	0.057	0.24	0.376	0.102
	Sensitivity	0.778	0.771	0.559	0.312	0.173
	Specificity	0.773	0.318	0.807	0.884	0.829
	Accuracy	0.777	0.342	0.783	0.78	0.763

Figure 2: LSTM Classifier evaluated on all datasets.

The f1 performance for the LSTM model was largely only successful on Twitter Loose, but was quite successful here, outperforming both best models from Davidson et al. (2017), which used the same data but in a tri-class problem, and Badjatiya et al. (2017), who used similar data.

4.2 Extensions

In order to effectively perform this analysis, we first took steps to handle class imbalance in our data, added some pre-processing steps to normalize text across datasets and implemented another strong classifier (a BERT-based classifier).

4.2.1 Imbalance Handling

In Milestone 2, we found that all of our datasets were significantly imbalanced, which resulted in low f1 scores for some baseline models. We implemented a variety of techniques to attempt to counter this imbalance. Ultimately, we found that these techniques often did more harm than good, and so opted not to use them in the final models.

Undersampling For datasets with over 10,000 majority-class samples, we randomly sampled 10,000 to become the new majority class. We tried to use this sparingly, as our datasets were already small and we did not want to lose too much data.

Text Augmentation Given our relatively small datasets, we focused more on oversampling the minority class than undersampling the majority. We did this via text augmentation, in which parts of a sentence are altered slightly to create a new sentence with a generally similar meaning. The goal of this process was to create new minority class examples that provided additional information (unlike simple oversampling) and made the models more robust to small variations in sentences. We tried a number of different augmentation schemes (including combinations of multiple strategies) and also varied the amount of extra samples we generated, but had the most success with a scheme that replaces words with neighbors in the embedding space, with a constraint to ensure their cosine similarity is at least 0.8. We also tried simple oversampling, where we simply added randomly sampled duplicates of existing minority examples.

Loss Function Modification Focal loss is an extension of cross entropy loss and it uses modulating factors like alpha and beta to focus training on hard to classify examples. Since our dataset is imbalanced, we decided to try Focal loss as the loss function to optimize.

$$\mathrm{FL}\left(p_{\mathrm{t}}\right)=-\alpha_{\mathrm{t}}\left(1-p_{\mathrm{t}}\right)^{\gamma}\log\left(p_{\mathrm{t}}\right)$$

We tried this loss function on the LSTM model. And initialized alpha = 0.25 and varied gamma from 1 to 5, but there was no improvement in the F1 score of the dev datasets. The following table shows that there was no significant improvement in the F1 score when using focal loss instead of CR (Cross Entropy).

Dataset	Focal Loss F1	CR Loss F1
twitter loose	0.964	0.967
twitter strict	0.245	0.339
political	0.835	0.816
reddit	0.063	0.021
wikipedia	0.760	0.756

4.2.2 Text Preprocessing

The preprocessing pipeline included cleaning/removing emojis, replacing urls, userids, emails, phone numbers and getting rid of any line breaks that are present in different datasets. We evaluated the datasets with and without emojis and there was no difference in the results, so we decided to remove the emojis. The emojis are encoded in different formats across the datasets, so getting rid of them made sense. Since, there are different types of urls, userids in the dataset, we put them under a name entity i.e $< url >$,$< userid >$, etc. And removed any line breaks that were present in each datapoint for different datasets. These transformations slightly improved the performance on the dev set of datasets from all domains.

4.2.3 BERT

We used a pretrained BERT model from the Keras library as another strong classifier. We used a small BERT model with 4 hidden layers, a hidden size of 512, and 8 attention heads, to keep training times manageable. Larger BERTs did not seem to improve performance.

We experimented with different loss functions: First, cross-entropy loss, which due to the class imbalance, did not generally produce good results. Second, macro F1 loss, which is a loss function based on 1 minus a modified F1 score, where true/false positives/negatives are calculated as the sum of the predicted probabilities. This generally

worked better, but sometimes caused the model to get "stuck" only predicting positives.

$$TP = \sum y_{true} \cdot y_{pred}$$
$$TN = \sum (1 - y_{true}) \cdot (1 - y_{pred})$$
$$FP = \sum (1 - y_{true}) \cdot y_{pred}$$
$$FN = \sum y_{true} \cdot (1 - y_{pred})$$

Figure 3: Defintion of true/false positives/negatives used when calculating macro f1 loss

The model was individually tuned for each dataset, with the final hyperparameters for that dataset's model being the combination with the best validation f1 score.

BERT - Best Model Metrics

Training Dataset	Metrics	Evaluation Dataset: Twitter (strict)	Twitter (loose)	Political	Reddit	Wikipedia
Twitter (strict)	F1	**0.396**	0.121	0.109	0.273	0.105
	Precision	0.372	0.923	0.156	0.419	0.105
	Sensitivity	**0.423**	0.065	0.085	0.202	0.105
	Specificity	0.961	0.973	0.950	0.938	0.899
	Accuracy	**0.934**	0.121	0.865	0.804	0.819
Twitter (loose)	F1	0.107	**0.976**	0.370	0.361	0.107
	Precision	0.057	**0.980**	0.515	0.392	0.102
	Sensitivity	0.909	**0.972**	0.288	0.335	0.111
	Specificity	0.179	0.901	**0.970**	0.884	0.889
	Accuracy	0.216	**0.961**	0.903	0.784	0.811
Political	F1	0.088	**0.847**	0.430	0.312	0.108
	Precision	0.047	**0.956**	0.419	0.356	0.103
	Sensitivity	0.608	**0.761**	0.441	0.277	0.114
	Specificity	0.334	0.825	0.933	**0.889**	0.888
	Accuracy	0.348	0.772	0.885	0.778	**0.810**
Reddit	F1	0.134	**0.507**	0.308	0.416	0.118
	Precision	0.786	**0.947**	0.247	0.327	0.102
	Sensitivity	0.464	0.346	0.407	**0.573**	0.141
	Specificity	0.705	**0.904**	0.865	0.437	0.861
	Accuracy	0.692	0.440	**0.820**	0.708	0.788
Wikipedia	F1	0.112	**0.905**	0.352	0.385	0.128
	Precision	0.060	**0.950**	0.388	0.391	0.103
	Sensitivity	0.875	**0.863**	0.322	0.378	0.168
	Specificity	0.251	0.777	**0.945**	0.869	0.836
	Accuracy	0.283	0.849	**0.883**	0.780	0.765

Figure 4: BERT Classifier evaluated on all datasets.

When evaluating on the same test set of the same data source used to train, BERT outperformed LSTM on F1 score for all data sources other than Wikipedia, where they performed essentially equally. Both LSTM and BERT performed significantly better than Naive Bayes and the simple baselines.

Despite being the best model we tried, it is worth noting that BERT was not necessarily successful across all datasets. Namely, while the accuracy was often relatively high, the f1 was quite low for all but Twitter (Loose), suggesting that it struggled to handle the signifcant class imbalance for most data sources. Notably, including offensive language as hate-speech (as in Twitter Loose) creates a far easier problem than excluding it (Twitter Strict), likely because the model has trouble more trouble differentiating offensive language from hate-speech than offensive language from inoffensive language.

Unfortunately, our efforts to mitigate class imbalance, such as text augmentation and loss function modification, were ultimately not successful, perhaps indicating that the problem itself is simply extremely noisy for most datasets.

Of course, for Twitter Loose, the model performed very well, again outperforming both best models from Davidson et al. (2017) and Badjatiya et al. (2017).

4.2.4 Transferability Analysis for Same-Platform corpuses

Same-Platform transferability refers to how well a model performs on the Twitter datasets when trained on one of the other Twitter datasets. Ultimately, the transfer between Twitter (strict) and Twitter (Loose) is of little interest, as they are the same dataset with different annotation schemes. It makes sense that those models would transfer poorly, as they are training to classify a scheme explicitly different from evaluation. Rather, of more interest is the transferability between the general Twitter sets and the Political set. Generally, Twitter (strict) and Political did not transfer well to each other, perhaps because the annotation scheme of the Political dataset leaned on the side of including offensive language in hate-speech. The comparison of Political and Twitter (Loose) is a bit more of a nuanced story. The test f1 of the model trained on Political and tested on Twitter (Loose) was .847, which is fairly decent. This means it performed 86.78% as well as when trained on Twitter (Loose) While the test f1 score of the transferred model trained on Twitter (Loose) and tested on Political was undoubtedly bad (.370), it actually still performed 86.05% as well as when trained on Political. Simply, the f1 of the transferred model was bad because even the f1 on the model trained on Political was bad. These results suggest that, when the platform or "structure" of the data remains the same, models can be transferred with relatively minimal performance degradation. Furthermore, the performance degradation was roughly the same regardless of which dataset was used for training and which was used for testing. Finally, the fact that this seemed to work for Twitter (Loose) but not Twitter (Strict) seems to suggest the need for parity in annotation schemes- if the two datasets are not in general agreement on what it means to be hate-speech, the model will not transfer. These results should be taken with a grain of salt- after all, the test performance is still ultimately poor when evaluating on Political, even if that may be more due to the model's inability to classify the Political

data in general rather than an inability to transfer.

It is also worth noting that the BERT model had far more success transferring than the LSTM, which achieved 74% of original performance when training on Twitter (Loose) and testing on Political, and an awful 22.88% of original performance when trained on Political and tested on Twitter (Loose). Perhaps the use of transformers and attention allowed BERT to get a better sense of context for each word, which allowed it to transfer more freely across domains.

4.2.5 Transferability Analysis for Cross-Platform corpuses

Next, we examined how the same BERT model might transfer when not only he content, but the structure of the text changes. To do this, we examined how models transferred between the datasets using Twitter as a base with datasets using Reddit and Wikipedia comments.

The transfer performance here is once again difficult to evaluate, as the normal performance on the Reddit and especially Wikipedia datasets were already quite low. This is especially problematic for the Wikipedia dataset, where the non-transfer f1 is .128, which means when transferring we may expect to see floor effects, where performance might not degrade much simply because it cannot get much lower. Indeed, we do see that transfer performance from the Twitter datasets on Wikipedia generally lies around 84% of the non-transfer performance, but it is hard to conclude if this is due to transferrence or floor effects. However, more interestingly, when training on Wikipedia on evaluating on Twitter (Loose), we achieve an f1 of .905, which is 92.73% the original .976. This is quite impressive, especially considering that the model trained on Wikipedia comments had far more success predicting hate-speech in general tweets than it did its own Wikipedia comments. It also transfers to the Political dataset nearly as well as Twitter (Loose) did, with an f1 of .352, or 81.86% of the performance when trained on Political. The Reddit dataset seemed to have a bit more trouble transferring, achieving an f1 of .308 (71.63%) on Political and .507 (51.95%) on Twitter (Loose). However, when trained on Twitter Loose and evaluated on Reddit, the model transferred better, acheiving a (still poor) f1 of .361 (86.78%). Training on Wikipedia led to almost no performance degradation on Reddit, with an f1 of .385 (92.55%), perhaps because the structure of Reddit and Wikipedia comments were closer to each other than to that of tweets, or perhaps because the Wikipedia provided a much larger training set. Overall, while it is harder to discern whether there is a strong bi-directional transference between datasets of different structure and context, it seems transference is possible.

4.2.6 Equalizing Dataset Sizes

Given that the Political dataset was far smaller than the others and the Wikipedia dataset was far larger, it was difficult to tell if success in transference was due to the content/structure of the data or simply because some datasets provided more training data. To equalize this factor, we attempted to undersample our Twitter (Loose) dataset down to the size of Political (sampling both training and validation sets so that they had the same number of positive and negative class instances as the Political counterparts), oversample Political to equal the size of Twitter (Loose), and undersample Wikipedia to equal the size of Twitter (Loose).

BERT - Best Model Metrics

Training Dataset	Metrics	Evaluation Dataset				
		Twitter (strict)	Twitter (loose)	Political	Reddit	Wikipedia
Twitter (loose) undersampled to Political	F1	0.062	0.319	0.000	0.002	0.008
	Precision	0.041	1.000	nan	0.500	0.121
	Sensitivity	0.125	0.190	0.000	0.001	0.004
	Specificity	0.840	1.000	1.000	0.999	0.997
	Accuracy	0.804	0.326	0.902	0.818	0.897
Political oversampled to Twitter (Loose)	F1	0.908	0.098	0.179	0.308	0.183
	Precision	0.832	0.051	0.098	0.182	0.101
	Sensitivity	1.000	1.000	1.000	1.000	1.000
	Specificity	0.000	0.000	0.000	0.000	0.000
	Accuracy	0.832	0.051	0.098	0.182	0.101
Wikipedia Undersampled to Twitter (Loose)	F1	0.098	0.908	0.179	0.311	0.181
	Precision	0.051	0.832	0.098	0.184	0.101
	Sensitivity	1.000	1.000	1.000	0.998	0.883
	Specificity	0.000	0.000	0.002	0.019	0.116
	Accuracy	0.051	0.832	0.100	0.197	0.193

Figure 5: BERT Classifier trained on under/over sampled datasets and evaluated on all datasets.

Unfortunately, in all scenarios, this process essentially ended up breaking the model, creating consistent specificity of 0. Thus, even in situations where f1 was high, the model simply was not predicting any items as non-hate-speech. This could speak to the importance of data size to the ability to transfer, but ultimately makes it difficult to form solid conclusions.

4.2.7 Multiple Datasets Experiment

4.2.7.1 Three Datasets Combined

With mildly promising results, we wanted to see if we could improve transferability by increasing the number of contexts on which the model is trained. First, we ran four experiments each with three combined datasets as the train dataset

and the other as the evaluation dataset. For example, we combied General Tweets (loose), Political Tweets, and Reddit Comments as the train dataset, and evalauted on the Wikipedia Comments. Here, we stop including Twitter (strict), as the annotation scheme clearly makes it unable to transfer well with the other datasets.

BERT - Multiple Datasets - dropout = 0.2, lr = 1e-5					
Training Dataset	**Metrics**	**Evaluation Dataset**			
		Twitter	Political	Reddit	Wikipedia
Political & Wikipedia & Reddit	F1	0.000			
	Precision	NaN			
	Sensitivity	0.000			
	Specificity	1.000			
	Accuracy	0.168			
Twitter & Wikipedia & Reddit	F1		0.387		
	Precision		0.529		
	Sensitivity		0.305		
	Specificity		0.970		
	Accuracy		0.905		
Twitter & Political & Wikipedia	F1			0.365	
	Precision			0.374	
	Sensitivity			0.355	
	Specificity			0.868	
	Accuracy			0.775	
Twitter & Political & Reddit	F1				0.083
	Precision				0.100
	Sensitivity				0.070
	Specificity				0.930
	Accuracy				0.843
Simple Majority Baseline	Accuracy	0.888	0.783	0.78	0.455

This strategy only proved to be the top transferer on the Political dataset, suggesting that providing multiple contexts may be useful sometimes, but not always. Overall, the f1 scores across the board were still quite low, meaning once again we must take this analysis with a grain of salt.

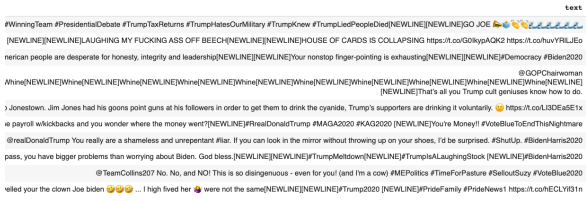
4.2.7.2 Four Datasets Combined

We also evaluated the combination of all datasets for training, with the goal of seeing if adding extra context could improve performance vs training and testing on the same set, potentially by adding extra data and forcing the model to be more robust. The learning rate and dropout parameter used for the training process was 1e-5 and 0.4 respectively. The combined model ultimately performed similarly to the models trained only on the dataset being evaluated, but did not improve performance for any of the cases, suggesting that the addition of contexts may not be a viable strategy in improving performance of hatespeech detection.

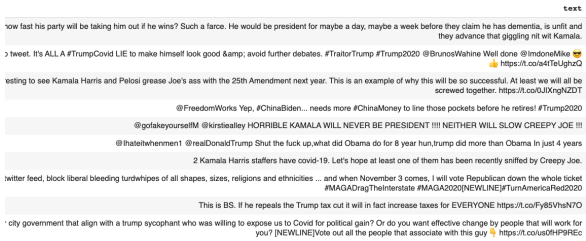
BERT - All Datasets - lr = 1e-5, dropout =0.4					
Training Dataset	**Metrics**	**Evaluation Dataset**			
		Twitter	Political	Reddit	Wikipedia
Twitter (loose)+ Political + Wikipedia + Reddit	F1	0.967	0.352	0.382	0.128
	Precision	0.975	0.418	0.372	0.101
	Sensitivity	0.959	0.305	0.392	0.173
	Specificity	0.881	0.953	0.852	0.827
	Accuracy	0.946	0.89	0.768	0.761

4.3 Error Analysis

Some false positive examples we got from the BERT model trained with Political Tweets and evaluated on Political Tweets are shown in the following table. These wrongly classified examples mainly contain curse/negative words and might be classified as hate-speech more easily.



Some false negative examples are shown in the following table. These examples contain less curse/negative words, and some use acronyms for curse/negative words.



Short texts are classified more accurately.

text	target	predicted	error	text_length
@ajplus Do not elect trump!	0	False	0	27
@JoeBiden SHUT IT #MAGA2020 🇺🇸	0	False	0	30
END the Left! NOW ! #TRUMP2020	0	False	0	30
@ThunderB Won't stop him. #Trump2020	0	False	0	36
@JoeBiden Yeah, LETS! #TrumpPence2020	0	False	0	37
@AmericanVoR NOT HERE #BidenHarris2020	0	False	0	38
@JoeBiden I'm in, Rev. #BidenHarris2020	0	False	0	39
Trump must have stock in Regeneron 🤦🏿	0	False	0	40
lagicJohnson Is True !! #trump2020 #AmericaFirst	0	False	0	49
sex? NOPE. Please win Biden #BidenHarris2020	0	False	0	50

Long texts produce more errors possibly due to more words lead to higher changes of confusion for the model.

text	target	predicted	error	text_length
@GOPChairwoman EWLINE]Whine[NEWLINE]Whine[NEWLINE]Whine[NEWLINE]Whine[NEWLINE] [NEWLINE]That's all you Trump cult geniuses know how to do.	0	True	1	536
LINE]CC: PRESIDENT JOE BIDEN[NEWLINE]CC: VP KAMALA HARRIS🇺🇸🇫🇷❤️	0	False	0	400
Hope you have snacks! 🛥️[NEWLINE]#BidenHarris2020 https://t.co/uTWH5XJ1LE	0	False	0	393
ates [NEWLINE]#Debates2020 [NEWLINE]#BidenHarris2020 [NEWLINE]🇺🇸🗳️🛥️	0	False	0	373
est criminals," ...[NEWLINE][NEWLINE]@BuzzFeed[NEWLINE]#BidenHarris2020	0	False	0	371
.INE]#Resist #Biden2020 #CORRUPTDOJ #TraitorTrump https://t.co/3hWfqAavsp	1	False	1	362
WLINE][NEWLINE]If you agree, RT this and SUBSCRIBE https://t.co/XZvWCzieik	0	False	0	360
...[NEWLINE][NEWLINE]#MAGA #Trump2020 #KAG2020 https://t.co/1fVqXJHj0J	0	False	0	359
alDisgrace [NEWLINE]#TrumpIsALoser[NEWLINE]#Vote https://t.co/lzZFKfCDOe	1	False	1	358
Landslide2020 [NEWLINE]#BidenHarrisToSaveAmerica [NEWLINE]#TrumpFailed	0	True	1	352

5 Conclusions

In this paper, we attempted to train BERT models to detect hate-speech across data from a variety of social networks. Ultimately, we achieved state-of-the-art performance on a dataset of general tweets, but failed to adequately handle the massive class

imbalance in our other datasets. While the accuracy for these other datasets was often high, the f1 scores were middling if not awful, likely due to the immense skew present in each dataset combined with the difficulty augmenting hate-speech in a way that provides new information while still representing natural hate-speech. BERT still outperformed both simple baselines as well as Naive Bayes and LSTMs in this task, but ultimately did not produce satisfactory results. Indeed, particularly when examining contexts like politics or Wikipedia comments (both of which can often be quite argumentative, passionate, and opinionated), it can be difficult to determine the line between impolite and hate-speech. We also examined how well models trained on certain datasets can transfer to others, finding that, in many cases, they actually did seem to transfer relatively well. This was particularly apparent when the "structure" of the text aligned, but could be seen, at least potentially unidirectionally, even when the structure differed, such as between tweets and Wikipedia comments. Notably, BERT transferred far more effectively than LSTM, perhaps representing a significant advantage for transformer-based architectures in training models for situations in which no annotated data is available. Ultimately, the analysis is held back by the fact that the f1 for all datasets other than the general tweets was poor, so it is difficult to know how much of the transfer performance was retained due to floor effects. Future literature should attempt to more effectively tackle class imbalance for individual models before re-assessing transferability. However, these results present a relatively promising outlook on detecting hate-speech even in burgeoning social network environments, where access to annotated data is limited or non-existent.

6 References

Vidgen, B., Nguyen, D., Margetts, H., Rossini, P., & Tromble, R. (2021). Introducing CAD: The contextual abuse dataset. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. https://doi.org/10.18653/v1/2021.naacl-main.182.

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

Schmidt, A., & Wiegand, M. (2017, April). A survey on hate speech detection using natural language processing. In Proceedings of the fifth international workshop on natural language processing for social media (pp. 1-10).

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In Proceedings of the international AAAI conference on web and social media (Vol. 11, No. 1, pp. 512-515).

Lara Grimminger and Roman Klinger (2020): Hate Towards the Political Opponent: A Twitter Corpus Study of the 2020 US Elections on the Basis of Offensive Speech and Stance Detection. 11th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (collocated with EACL 2021).

7 Appendicies

BERT - Hyperparameter Tuning - F1 Score							
Training Dataset	**Hyperparameters**		**Evaluation Dataset**				
	Dropout	**Learning Rate**	Twitter (strict)	Twitter (loose)	Political	Reddit	Wikipedia
Twitter (strict)	0.2	1.00E-05	*0.396*		0.110	0.273	0.105
		1.00E-03	*0.098*		0.179	0.308	0.183
	0.4	1.00E-05	*0.395*		0.086	0.237	0.096
		1.00E-03	*0.098*		0.179	0.308	0.183
Twitter (loose)	0.2	1.00E-05		*0.976*	0.370	0.361	0.107
		1.00E-03		*0.908*	0.179	0.308	0.183
	0.4	1.00E-05		*0.975*	0.340	0.385	0.117
		1.00E-03		*0.948*	0.213	0.323	0.175
Political	0.2	1.00E-05	0.092	0.835	*0.418*	0.340	0.104
		1.00E-03	0.098	0.908	*0.179*	0.308	0.183
	0.4	1.00E-05	0.088	0.847	*0.430*	0.312	0.108
		1.00E-03	0.098	0.908	*0.179*	0.308	0.183
Reddit	0.2	1.00E-05	0.098	0.873	0.245	*0.379*	0.154
		1.00E-03	0.098	0.908	0.179	*0.308*	0.183
	0.4	1.00E-05	0.134	0.507	0.308	*0.416*	0.118
		1.00E-03	0.098	0.908	0.179	*0.308*	0.183
Wikipedia	0.2	1.00E-05	0.113	0.904	0.375	0.386	*0.128*
		1.00E-03	0.098	0.908	0.179	0.308	*0.183*
	0.4	1.00E-05	0.112	0.905	0.352	0.385	*0.128*
		1.00E-03	0.098	0.908	0.179	0.308	*0.183*
Baseline – Always Predict Positive:			0.098	0.908	0.179	0.308	0.183

Figure 6: Full BERT Tuning Results

The color indicates performance relative to a simple baseline of predicting every sample as positive. Yellow indicates equal to the baseline, orange indicates slightly worse than the baseline, red indicates more than 20% worse than the baseline, light green indicates slightly better than the baseline, and dark green indicates more than 20% better than the baseline.