Automatic Detection of Hybrid Human-Machine Text Boundaries

Joseph Cutler **Liam Dugan** **Shreya Havaldar** **Adam Stein**
Trustees of the University of Pennsylvania
{jwc, ldugan, shreyah, steinad}@seas.upenn.edu

Abstract

Much effort has been spent in recent years on using machine learning to automatically generate text which passes as having been written by humans. Because of significant advances in this field, language models are able to generate very convincing text. However, a reader with a discerning eye can occasionally distinguish real text from fake. In this project we introduce a novel detection task format: detecting the boundary between human written prompt and machine generated continuation. We train various classifiers to predict not only the true boundary but the boundary as assessed by human annotators. We find that using pretrained embeddings outperform perplexity and pairwise sentence coherence based methods.

1 Motivation

Much effort has been spent in recent years on using machine learning to automatically generate text which passes as having been written by humans. Because of significant advances in this field, language models such as GPT-2 (Radford et al., 2019) or CTRL (Keskar et al., 2019) are able to generate very convincing text. However, a reader with a discerning eye can occasionally distinguish real text from fake. It is still not fully understood exactly what features human annotators use to make these predictions, but it is clear that they are not always the same features that classifiers use (Ippolito et al., 2020).

In this project, we utilize the "Real or Fake Text" dataset (Dugan et al., 2020) to investigate the aspects of generated text which make it more or less likely to be detected by a human observer as real or fake. The problem is structured as a classification task: Given a 10-sentence passage of text (which at some point transitions from being human-written to machine generated), predict the index of the last human written sentence (the "boundary sentence"). Since the dataset includes human annotations, we can choose to either predict the true boundary or to *predict the human predictions*. This second task may be even more interesting, as it may give us clues as to how humans and machines detect text differently. In addition, the dataset includes various metadata elements about each individual human annotator such as their major, familiarity with generated text, and familiarity with the domain from which the prompt was sampled, giving us even more information to use.

This task is particularly well suited to being solved by machine learning. To start, the task is inherently distributional in nature; generative models sample from their own predicted distribution of next word tokens at each time step. Thus, it makes sense for distributional features (such as word embeddings or perplexity) to perform well here. In addition, in the case of the true boundary detection task, we have an abundance of data; if we need more data for generated text detection we can always generate more (given a known generative model and sampling strategy). Finally, in the case of human annotator decisions, machine learning can give us insights into the importance of different factors in the decisions made by human annotators.

The outcome of this project will have multiple interesting implications. First, our results will shine light on how the well-known pitfalls of current language generation techniques affect the distinguishability of real text from fake text. Second, they will provide important feedback to the *designers* of language models, who can use our findings to iteratively improve the realism of their models' outputs. Finally, they can also be used to train people to be better at distinguishing real and fake text as they will provide insight into features which annotators may mistakenly assume to be signs of generated text.

2 Related Work

Much of the previous work done in the automatic detection of machine generated text has centered around the passage-level binary classification task. That is, given a passage, determine whether the entire passage was machine-generated or human-written. Work from Solaiman et al. (2019) showed that fine-tuning a RoBERTa model (Liu et al., 2019) on generations from a specific GPT-2 (Radford et al., 2019) model and specific decoding strategy can achieve over 99% accuracy on detecting output from the same model. Interestingly, they show that some classifiers are able to generalize to other decoding strategies and model sizes while oth-

ers are not. While this is an encouraging result, it is worth noting that the binary detection task is significantly easier than the boundary detection task; even a single token of prompting can significantly change the subsequent distribution of generated text and make automatic detection much harder (Ippolito et al., 2020). Thus performance on the boundary detection task should not be directly compared to the binary detection task, rather, they should be used in conjunction to better understand the benefits and limitations of certain detection methods.

Other previous baseline methods for the binary detection of generated text have considered features such as bag-of-ngrams (Zellers et al., 2020), histogram of token likelihoods (Gehrmann et al., 2019), and total probability of the sequence (Solaiman et al., 2019). However, work by Ippolito et al. (2020) and many others have shown that these methods are soundly defeated by fine-tuned BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models. We follow this and other previous work in primarily looking at these models as our main experiments. For our baselines we skip experimenting with bag-of-words and histograms and instead use total probability of sequence similarly to Solaiman et al. (2019) and Zellers et al. (2020). Other methods such as using higher-order n-gram statistics (Gallé et al., 2021) or Topological Data Analysis (Kushnareva et al., 2021) were also not attempted, but could be investigated in future work.

As for the human detection of generated text, Ippolito et al. (2020) has shown that humans are the most easily fooled by generated text that was sampled using the most probable token at each time step (nucleus sampling (Holtzman et al., 2020) with p=0.0). Paradoxically, this manner of sampling is the most detectable by automatic detection algorithms. Our results show that this trend still holds in the case of the boundary detection task.

3 Data

3.1 Data Collection

The dataset used for this project was gathered using the "Real or Fake Text" (RoFT) platform (Dugan et al., 2020). In this detection task, annotators are shown a passage of text one sentence at a time. The first several sentences are from a human-written text source and the next several sentences are a machine-generated continuation. The user's goal is to guess where the boundary between human-written and machine-generated text is. They are also asked to give an explanation for their choice, which can be a selection from a radio button set of options or a free-text response. Afterwards, the true boundary is revealed. Statistics on the distributions of annotations across different domains in the dataset can be found in Table 2.

This dataset was collected between late September 2021 and late October 2021 from undergraduate and masters students at Penn taking CIS521 "Introduction to Artificial Intelligence"[1]. Students were given extra credit proportional to the amount of annotations they got correct. In addition to this, a leaderboard of top scorers was visible on the publicly hosted RoFT website[2], further adding to students' motivation to perform well on the task. Students were also given a briefing document that helped to teach them how to detect generated text more effectively. This document included examples of common mistakes that generative models make as well as tips on how to spot them. After the students finished their annotation they were also given an exit survey that asked them their familiarity with the different domains of text that they had just annotated as well as their level of English fluency and their undergraduate major.

The full dataset can be viewed at the following link: https://drive.google.com/file/d/1YlG1O_w_fVTc9oLkoz9eBlmzHFuGfNze/view?usp=sharing

3.2 Features

For the task of detecting generated text we are only interested in the basic textual features: the text of the prompt, the generated continuation, and the true boundary. However, for the human prediction detection task one could utilize all of the provided features. For the curious reader, a table of all of the features can be found in Table 1

3.3 Preprocessing

A great deal of the data pre-processing was handled by the RoFT project authors. Because of their efforts, we started with a cleaned CSV file containing the 27501 observations, with the features listed in Table 1. Sentence boundaries within the text fields "prompt" and "generation" were marked with a "__SEP__" token, which we split on to construct vectors of sentences. We ensured that all of the paragraphs were at least ten sentences by filtering out any that were shorter: these are occasionally generated when the language model fails to generate more text. Finally, as is described in greater detail in Section 5, we used pre-trained word embeddings to turn our textual features into real-valued ones.

4 Problem Formulation

A formal definition of the two problems we will be tackling for this project are listed below.

True Boundary Prediction Given a context passage $C = \{s_0, ..., s_9\}$ where s_i is the ith sentence in the passage, the true boundary prediction task is to predict the index j corresponding to the first machine-generated sentence s_j. We assume that for all $1 \leq j \leq i$, s_i is machine generated and for all $k < j$, s_k is human-written.

[1] https://artificial-intelligence-class.org/

[2] http://roft.io

Feature Name	Description
prompt	Text of the prompt
generation	Generated text
annotator	Unique annotator ID
time	Date and time of annotation
model	Model used to generate text
dataset	Source of the prompt
dec_strategy	Nucleus Sampling parameter p
true_boundary	Index of last human sentence
pred_boundary	Annotator's guess of boundary
reason	Annotator's reason for answer
major	Annotator's major
familiarity	Annotator's domain familiarity
read_guide	Did annotator read the guide

Table 1: Features present in the RoFT dataset (Dugan et al., 2020). The features we used are bolded.

Domain	n	Models
News	4,957	GPT2-XL
Reddit	5,230	GPT2, GPT2-XL
Recipes	12,186	GPT2-XL, Finetuned GPT2-XL
Total	27,471	All Models

Table 2: Statistics on the distribution of human annotations collected across different domains in the RoFT dataset. News data was sampled from Sandhaus (2008). Reddit data was sampled from Fan et al. (2018). Recipe data was sampled from Marin et al. (2019).

Human-Predicted Boundary Prediction Similarly to the previous task, the human-predicted boundary prediction task models the input passage. Given a passage $C = \{s_0, ..., s_9\}$ where s_i is the ith sentence in the passage, the boundary prediction task is to predict the *average* index predicted by all of the annotators who were given the passage C. Figure 1 shows our dataset's distribution of average boundary values picked by annotators, and Figure 2 shows our dataset's distribution of human labels in comparison to the true labels. We see that the human labels are extremely noisy predictors of the true boundary, but there is small correlation between the average predicted boundary and the true boundary.

Loss Function Our loss function of choice is cross-entropy loss. During training, this penalizes our model for *any* incorrect prediction, high or low. One could alternatively use a "smoother" loss function such as distance (under some metric) from the correct boundary. While our testing indicated that this was the optimal choice, it seems likely that a smoother alternative would have benefits. Future work should investigate this possibility more thoroughly.

5 Methods

5.1 Baselines

Our first two baselines are a simple random baseline and a majority class baseline. We decided to use the majority class baseline to expose any class imbalances in our data and the random baseline to show the worst possible classifier. Our third baseline is a perplexity based detection model. We use the original language model that was used to generate the text and get the perplexity for each sentence in the passage conditioned on all the previous sentences. We then use these sentence-wise perplexity metrics to predict the true and human-predicted boundary using simple heuristic methods like the location of largest decrease in rolling average perplexity. We consider this method to be a baseline since we use the model used for producing the text to produce features, so we have more information than we truly should be given in this task.

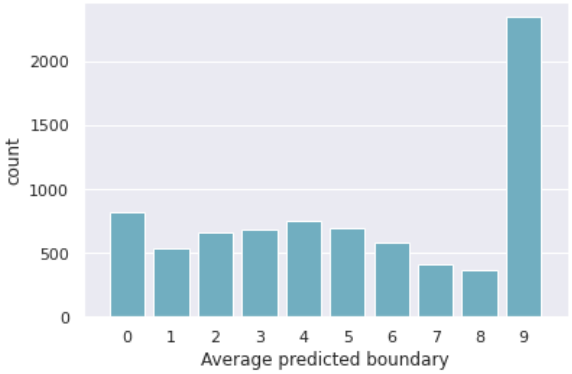

Figure 1: Distribution of Average Predicted Boundary Value across Annotators

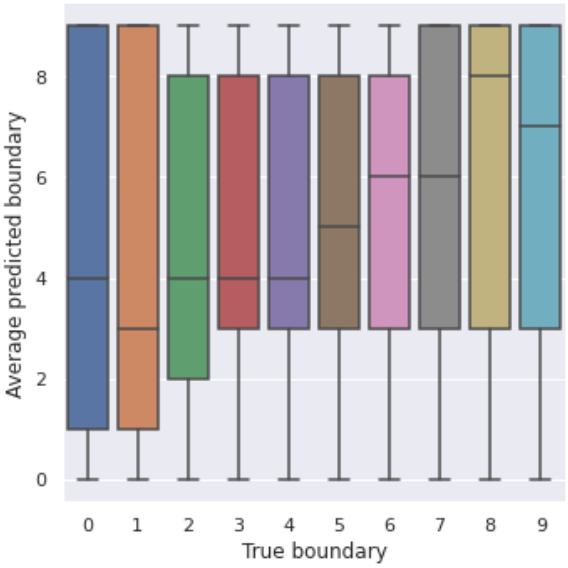

Figure 2: Distribution of average boundary label for each of the 10 true boundary indexes.

5.2 RoBERTa

Our first main method of comparison is to use the average RoBERTa (Liu et al., 2019) embedding of the words in each sentence as a sentence embedding. To do this we prepend the [CLS] and append the [SEP] token to

Domain	Model	Top p	Size (train/test)	Rand	Maj	Human	Perp	RoBERTa (OOD/ID)	SRoBERTa (OOD/ID)	STS (OOD/ID)
NYT	gpt2-xl	0.0	324/137	0.08	0.12	0.15	**0.33**	0.26/0.21	0.29/0.22	0.12/0.10
NYT	gpt2-xl	0.4	314/117	0.11	0.12	0.14	0.28	0.32/0.22	**0.34**/0.23	0.14/0.10
NYT	gpt2-xl	1.0	291/128	0.11	0.12	0.19	0.02	0.13/**0.41**	0.23/0.26	0.13/0.13
Reddit	gpt2	0.4	744/245	0.09	0.21	0.26	0.16	0.59/**0.62**	0.33/0.35	0.23/0.23
Reddit	gpt2-xl	0.0	136/55	0.12	0.13	0.10	0.31	0.29/0.23	**0.44**/0.15	0.20/0.05
Reddit	gpt2-xl	0.4	149/60	0.11	0.12	0.14	0.27	0.34/0.23	**0.42**/0.16	0.15/0.08
Reddit	gpt2-xl	1.0	134/43	0.14	0.14	0.23	0.04	0.15/0.15	**0.30**/0.21	0.07/0.09
Recipes	gpt2-xl	0.4	1111/379	0.10	0.50	0.35	0.09	0.59/0.60	0.59/**0.65**	0.56/0.56
Recipes	finetuned	0.4	1847/647	0.10	0.12	0.25	0.05	0.38/0.39	**0.43/0.43**	0.10/0.11
All	All	0.4	5092/2047	0.10	0.20	0.25	0.16	0.32 (0.93)	**0.42** (0.89)	0.22 (0.26)

Table 3: Results for the **true boundary detection task** for all data subsets using the highest performing classifier under each proposed method. Peformance on overall dataset is reported in the form test accuracy(train accuracy). "Out of Domain" (OOD) classifiers were trained on all available data and evaluated on just the subset. "In Domain" (ID) classifiers were trained only on the given subset. "Rand" = Random Guessing; "Maj" = Majority Class Baseline. "Perp" = Perplexity Baseline.

each sentence of the passage in isolation. After this we concatenate all of the sentence embeddings together into a 7,680 dimensional passage embedding and pass that into a logistic regression classifier with cross-entropy loss.

5.3 SRoBERTa

Our second method is the same as the previous method but instead of taking the average RoBERTa embeddings we use SRoBERTa (Reimers and Gurevych, 2019) to get a sentence embedding for each sentence in the passage. SRoBERTa is a RoBERTa model variant that was trained in such a way as to maximize the predictive power of the average of token embeddings. The sentence embeddings of this model have shown to be effective in a wide variety of tasks such as Natural Language Inference. Similarly to the previous method, we concatenate the sentence embeddings together and feed the passage embedding to a logistic regression classifier.

5.4 Semantic Textual Similarity

The third method is similar to the second method, however instead of taking the embeddings as input to the classifier, we take the pairwise cosine similarity of each sentence with each preceding sentence as input. This results in a feature vector of size 9. We then feed this into logistic regression.

For the extraction of RoBERTa embeddings we use the HuggingFace Transformers[3] library. For SRoBERTa we use the sentence_transformers package[4]. Finally, for running logistic regression and other classifier comparisons we use scikitlearn[5].

6 Experiments and Results

For evaluation we took the data and split it into three subsets (train, val, and test) with an 80%, 10%, 10% split. Because the data contains many duplicate entries

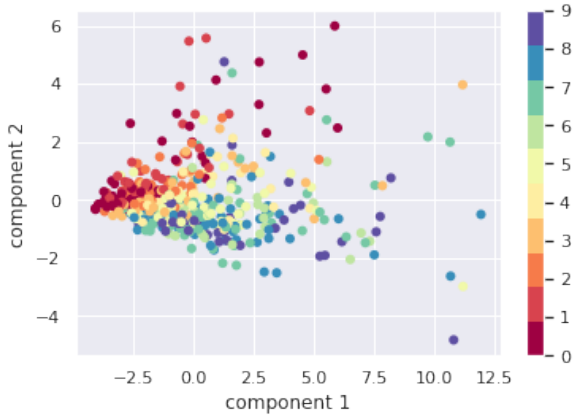

Figure 3: 2 component PCA of the Perplexity embeddings for NYTimes text using GPT2-XL for generation with $p = 0.0$. As expected for argmax sampling, perplexity clearly distinguishes between the different true boundary classes.

(each row of our dataset corresponds to one annotator's annotation and multiple annotators can annotate the same example of generated text), we made sure to deduplicate each set individually and ensure that there was no leakage between the datasets.

As an accuracy metric, we chose to use a standard 0/1 error classification accuracy, because it mirrors our choice of cross-entropy loss as our loss function: we only take into account exactly-correct predictions, and slightly incorrect ones are as good as very incorrect ones.

All of of our models are given the same random seed (42069) for reproducibilty purposes, and we use the SAGA solver (Defazio et al., 2014) for optimization.

Once the models were decided upon and the hyperparameters were tuned using the validation set, we ran all of our experiments on the test set. All of our experiments were performed twice: once after the models were trained to predict the true boundary indices on the test set, and then again when the models were trained to

[3]https://github.com/huggingface/transformers
[4]https://github.com/UKPLab/sentence-transformers
[5]https://scikit-learn.org/

Domain	Model	Top p	Size (train/test)	Rand	Maj	Perp	RoBERTa (OOD/ID)	SRoBERTa (OOD/ID)	STS (OOD/ID)
NYT	gpt2-xl	0.0	324/137	0.09	0.23	0.08	0.18/0.18	0.28/**0.29**	0.23/0.20
NYT	gpt2-xl	0.4	314/117	0.11	0.22	0.08	0.24/0.14	0.18/0.21	0.28/**0.29**
NYT	gpt2-xl	1.0	291/128	0.11	0.19	0.09	0.15/**0.21**	0.13/0.14	0.15/0.18
Reddit	gpt2	0.4	744/245	0.10	0.21	0.08	0.15/0.15	0.22/**0.23**	0.22/0.19
Reddit	gpt2-xl	0.0	136/55	0.11	0.28	0.05	0.17/0.30	0.2/**0.42**	0.20/0.22
Reddit	gpt2-xl	0.4	149/60	0.09	0.20	0.08	0.15/0.23	0.18/**0.28**	0.23/0.27
Reddit	gpt2-xl	1.0	134/43	0.11	0.21	0.10	0.15/**0.31**	0.14/0.12	0.26/0.19
Recipes	gpt2-xl	0.4	1111/379	0.10	0.20	0.06	0.13/0.10	0.39/**0.40**	0.18/0.17
Recipes	finetuned	0.4	1847/647	0.10	0.21	0.07	0.20/0.193	0.27/**0.27**	0.20/0.20
All	All	0.4	5092/2047	0.10	0.21	0.06	0.17 (0.86)	**0.25** (0.95)	0.22 (0.27)

Table 4: Results for the **human-predicted boundary detection task** for all data subsets using the highest performing classifier under each proposed method. Peformance on overall dataset is reported in the form test accuracy(train accuracy). "Out of Domain" (OOD) classifiers were trained on all available data and evaluated on just the subset. "In Domain" (ID) classifiers were trained only on the given subset. "Rand" = Random Guessing; "Maj" = Majority Class Baseline. "Perp" = Perplexity Baseline.

Experiment	Parameter	Search Space	Optimal Value
RoBERTa	Regularization Penalty	[1.0,0.1,0.01,0.001,0.005,0.0001]	0.001
SRoBERTa	Regularization penalty	[0.0001, 0.0005, 0.001, 0.005] ['l2', 'l1']	0.0005 l2
STS	Regularization penalty	[0.01, 0.05, 0.1, 0.5] ['l2', 'l1']	0.5 l2

Table 5: Hyperparameter Tuning across Experiments: Values Searched and Value Selected

predict the (average) *human-predicted* boundary indices. These results are reported in full in Table 3 and Table 4, respectively.

6.1 Baselines

The three baseline methods were random, majority label, and a non-parametric perplexity based detection model. As expected, the random baseline got roughly 10% accuracy across all the data subsets and on the full dataset. More interesting is the majority class baseline which exposed the large class imbalances present in the dataset. In particular, the subset of the true boundary data for the Recipes subset which was generated with GPT2-XL using a top-p sampling with $p = 0.4$ had 50% of the samples labeled with the true boundary as 9. For the human predicted labels, we see that overall 21% of our data is labelled with 9. A model which simply outperforms random guessing may still not be a valuable model if it does not outperform the majority baseline since we want to learn a model which can detect the true or predicted boundary better than always guessing 9.

We found that the performance of the perplexity baseline was highly dependent on the top-p sampling method used for generating the text. For text generated with GPT2-XL using p=0.0, the performance of the perplexity baseline is roughly twice as good as the majority baseline. We see in Figure 3 the extent to which perplexity is able to distinguish between the different classes for $p = 0.0$. However, as the value of p increases, the performance of the perplexity baseline decreases since the generated text becomes less predictable and may begin to look closer to the true distribution of human sentences. This goes to an extreme when $p = 1.0$ since the machine generated text has even higher perplexity than the human generated text. This causes the perplexity to increase for machine generated sentences, so we see that the performance is worse than random guessing since our detector will not predict the boundary where perplexity increases.

Since the non-parametric perplexity baseline performed so well, it may seem like we should train a more sophisticated parametric model on these features. We decided against this approach because this assumes prior knowledge of the generative model. For our problem definition we did not allow for this. One possible extension of this to the case where we don't a priori know the model is to compute the perplexity scores for all popular pre-trained models for each sample and learn a model over those features.

6.2 RoBERTa

In order to determine the optimal model for classification using concatenated RoBERTa embeddings, we tried a variety of models, and calculated their accuracy on the validation set. These models and their best validation accuracies (over coarsely tuned hyperparameters) are included in Table 6.

Among these models, logistic regression has the highest accuracy, and so we chose to focus on it for our experimentation. We then ran a more fine-grained hyperparameter search (more values checked) on the regression coefficient of the logistic regression to maximize our validation accuracy. The results of this search, as well as the hyperparameter values we searched over, can

Model	Validation Accuracy
Ridge Regression	0.369
Logistic Regression	0.377
KNN	0.253
Neural Network	0.198
AutoML	0.104

Table 6: RoBERTa Model Experimentation for the true boundary prediction task

be found in Table 5.

6.3 SRoBERTa

Similarly to the procedure employed for the RoBERTa embeddings, we experimented with a variety of SKLearn classifiers and calculated accuracy on the validation set. The models and their resulting accuracies are included in the table below. Manual hyperparameter tuning was used for each of these tested models, with a more involved hyperparameter search on the highest performing model to create our final SRoBERTa classifier.

Model	Validation Accuracy
Random Forest	0.2154
Logistic Regression	0.4180
Multiclass SVM	0.2291
KNN	0.2706
Neural Network	0.3249

Table 7: SRoBERTa Model Experimentation for the true boundary prediction task

Based on the results, we selected the logistic regression classifier. We then performed more thorough hyperparameter tuning in order to maximize validation accuracy. The parameters tuned and range of values tried can be found in Table 5.

6.4 Semantic Textual Similarity (STS)

In order to determine the optimal model for classification using our semantic textual similarity features, we conducted the same experimentation process. The SKLearn models we tried and their resulting accuracies are included in Table 8. Manual hyperparameter tuning was used for each of these tested models, with a more involved hyperparameter search on the highest performing model to create our final STS classifier.

Model	Validation Accuracy
Random Forest	0.2154
Logistic Regression	0.2184
Multiclass SVM	0.2120
KNN	0.2052
Neural Network	0.1411

Table 8: STS Model Experimentation for the true boundary prediction task

We noticed very comparable accuracies between the random forest classifier and the logistic regression classifier. Ultimately, we decided to go with the logistic regression classifier as the training time needed for the model was shorter. We then performed more thorough hyperparameter tuning in order to maximize validation accuracy. The parameters tuned and range of values tried can be found in Table 5.

6.5 Evaluation and Subset Testing

After model selection and hyperparameter tuning, we used our resulting classifiers (RoBERTa, SRoBERTa, STS) to calculate how well our model performed on the test set. Table 3 shows our performance on classifying the true boundary, and Table 4 shows our performance on classifying the human-predicted boundary.

In order to analyze which subsets of data our model performed the best/worst on, we filtered by how the machine-generated text was created (gpt2, gpt2-xl, fine-tuned), which value of the Nucleus Sampling (Holtzman et al., 2020) parameter p was used (0.0, 0.4, 1.0), and what the source of the human-written text was (New York Times, Reddit Short Stories, Recipes). For each subset evaluation, we trained our model both in-domain (training on only the data points from the subset we test on) and out of domain (training on the entire training set). The resulting test accuracies for each subset can be found in Table 3 on the true boundary, and Table 4 on the predicted boundary. These tables also contain calculated accuracies for each subset using our baseline classifiers, random, majority class, and perplexity.

7 Conclusion and Discussion

Findings and Model Comparisons While the boundary prediction task is clearly a very difficult one (none of our classifiers consistently performed better than 50% accuracy), we do have many worthwhile takeaways from this project. For starters, we see that the embeddings based classifiers significantly outperform human accuracy on spotting generated text. This is reassuring and is a good sign that classifiers for generated text are not only useful but necessary going forward.

In the case of the human-predicted boundary prediction task, we see that the majority class baseline is very strong - outperforming all other methods on our limited size dataset when aggregating over all subsets. The accuracy of the majority class baseline indicates that the distribution of sentences selected as the boundary by human annotators is highly non-uniform. Further research should investigate the psychological factors that may influence annotators to select certain sentences.

We also see that the performance of the perplexity baseline does very well for p=0.0 but worse than chance for p=1.0 when predicting true boundary. This is consistent with what we would expect, given that text produced by p=0.0 will have artificially high likelihood as determined by the language model from which the text was generated. Interestingly, perplexity does identical

to random chance at the human-predicted boundary detection task – indicating that human annotators do not at all look at perplexity for their predictions. In addition, similar to the results from (Ippolito et al., 2020), we see that human accuracy at predicting the true boundary is lowest for text produced by p=0.0 while this is the easiest for the perplexity baseline to detect.

Finally, we see that, in the case of the Semantic Textual Similarity (STS) experiment, while the embeddings significantly outperform STS for the true boundary detection, the human-predicted boundary prediction task tells a different story. While the SRoBERTa classifier is still the best overall, the margin is significantly smaller, with STS outperforming the standard RoBERTa embeddings and perplexity. From this we can tentatively draw the conclusion that human annotators are particularly sensitive to sudden sentence-level semantic shift and consider text that does this to be unnatural.

Lessons Learned The main lessons we learned from this project were about dealing with data. There were many times where we realized that our classifiers were conditioning on bad features or other artifacts that we left in the data. Ensuring there was no leakage between training and test data, ensuring that we had proper boundary values, and ensuring that we dealt with NaNs properly were all sources of unexpected complications. We realized time and time again that the hardest part of a Machine Learning project is properly cleaning the data and that the easier part is writing the code to run the model.

Another lesson we learned is that the most complex classifier is not always the most optimal one. In our case, given a smaller dataset the best performing classifier was, in fact, the standard Logistic Regression classifier and not any other more powerful method. This reaffirms the importance of testing classifiers that are generally considered to be less powerful and not always assuming that a neural network will be the highest performer.

Future Work Finally, future work should seek to further experiment with not only the methods chosen but also the problem formulation. We chose to use the standard 0/1 accuracy with cross-entropy loss due to our framing of the task as a classification task. However, this is not the only framing of this task that is possible – one could make the case that a predicted boundary that is one sentence away from the true boundary is a better prediction than one that is much further away. Using a loss function such as the L2 distance of the prediction away from the boundary may have led our classifiers to perform better.

On top of this, while we were limited to only selecting data that was annotated by humans for our human-predicted boundary prediction task, this is not the case for the true boundary prediction task. It is theoretically very easy to generate extra data for this task and future work should seek to utilize much more data when evaluating classifiers on true boundary prediction.

References

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Liam Dugan, Daphne Ippolito, Arun Kirubarajan, and Chris Callison-Burch. 2020. Roft: A tool for evaluating human detection of machine-generated text.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation.

Matthias Gallé, Jos Rozen, Germán Kruszewski, and Hady Elsahar. 2021. Unsupervised and distributional detection of machine-generated text.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. Gltr: Statistical detection and visualization of generated text.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Laida Kushnareva, Daniil Cherniavskii, Vladislav Mikhailov, Ekaterina Artemova, Serguei Barannikov, Alexander Bernstein, Irina Piontkovskaya, Dmitri Piontkovski, and Evgeny Burnaev. 2021. Artificial text detection via examining the topology of attention maps.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2020. Defending against neural fake news.