Learning to Trick Humans: Evaluation Criteria for Human-Written and Machine-Generated Text

Arun Kirubarajan*
University of Pennsylvania
kiruba@cis.upenn.edu

Liam Dugan*
University of Pennsylvania
ldugan@seas.upenn.edu

Abstract

Large neural network based language models (1B+ parameters) such as GPT-2 have been shown to produce high-quality text when trained on large datasets (Radford et al., 2019). However, these models still make errors and there lacks a set of criteria to diagnose issues in natural language generation. Furthermore, the rise of AI-generated content on the internet prompts the question of how good humans are at detecting computer-generated text. In this research, we present a Turing Test inspired task for humans to annotate evaluation criteria by discriminating between human and machine written content. By presenting sentences one at a time to human annotators, we show that text generation machines can make it an average of 2.14 sentences before a human reader notices the text has switched to a machine. We also create an error taxonomy of generated text by having annotators diagnose poor text with error tags (e.g. grammar, entailment, common-sense).

1 Introduction

There is little agreement on how to evaluate natural language generation systems using human annotation due to the wide variety of existing standards (van der Lee et al., 2019). This experiment will be focused on using human annotation to answer what qualitative and linguistic qualities in particular give away the fact that a text is or is not human-written. Annotators will be able to select from a predefined list of error categories, and select sentences that provide the evidence for the annotator distinction. For example, an annotator may notice a failing coreference resolution, and provide the sentence containing the original named entity as context for the error. The goal of this project is to obtain qualitative human assessment of provided text in a systemized fashion in order to increase the accuracy and "human"-ness of text generation.

2 Related Work

Previous work has shown that properly evaluating natural language generation systems automatically is difficult (van der Lee et al., 2019). Various methods for automatic and human evalution have been proposed over time to identify weaknesses in generation systems.

The bAbI dataset (Weston et al., 2015) aims to diagnose a model's limitations by providing the particular granular tasks for which it fails. This work was a major inspiration for ours as we aim to develop a system of categories for annotators to tag generated sentences with as a means of analyzing a model's capabilities and the text's overall linguistic quality. The ChatEval framework (Sedoc et al., 2019) also provides a suite of automatic metrics but also includes studies of human annotator's preference between two given generation systems to allow for more human-based grounding of their metrics.

Additionally, previous work done by Ippolito et al. (2019) has analyzed how choices in sampling strategy and differing lengths of prompts contribute to the accuracies of both human and machine systems in specifically the detection of machine generated text.

This paper aims to provide three other dimensions to this previous work:

1. To assess the distribution of qualitative linguistic qualities that humans use to identify generated texts as they vary with prompt length.

2. To identify the relative speeds (e.g. number of sentences) that it takes before a human can determine that something is machine generated

3. To investigate which qualitative errors tend to occur specifically at these boundaries between human and machine text.

3 Data

In order to minimize the influence of factual errors in favor of more semantic errors, we sampled our prompts from the dataset that was used to train the popular text adventure game AI Dungeon[1]. The dataset consists of over 100k lines of user-submitted text excerpts from text-based fantasy stories posted to the website Choose Your Story[2].

Below is a sample from our training data:

```
"You, you and you!"
he shouts,
selecting three rats
from the group.
"Start moving the
prisoners back."
"Yes, yes," they snarl,
as they
begin to shout and
whip the prisoners
into movement.
"You!" he shouts, pointing
a long finger at you.
"Kill the horses."
You nod, walking
over to the horses.
```

Using this data allowed us to prevent annotators from using outside knowledge to assist in discrimination while additionally allowing annotators more room to focus on particular details of text, such as awkward wording, contextual errors, and co-reference mistakes. One issue with this data was the frequency of dialogues turns and the ambiguity this creates when deciding what is and isn't a sentence. We overcame this by specifically employing a sentence tokenizer (See Method section) instead of assuming period or newline would be a good split. Additionally we had to deal with <|startoftext|> and <|endoftext|> tokens and the occasional human grammatical error. However, these difficulties were rare and overall we are very pleased with the use of this data.

4 Method

For our error taxonomy we maintain four core categories of errors: Entailment Errors, Grammatical Errors, Common-Sense Errors, and Repetition Errors. We believed each to be distinct from the others, while representing a common tendency for poor model generations.

4.1 Annotator Interface

The annotation software was built in full-stack Python using the Django Framework and a SQLite database. The choice of a full relational database was to ensure that sophisticated queries could be made on the collected annotations for analysis (e.g. average number of read sentences for a specific prompt across all users). The Django Framework contains libraries for a various functionalities of production-ready application servers, and reduced overall development time of the system. The error taxonomy form shown to our annotators is shown in Figure 2.

4.2 Prompt Length

One of the most important considerations taken in this project was to ensure that our annotators were not conditioned to look for errors in otherwise inconspicuous human-written text. Our main goal was to pinpoint sentences in text generations where it was suddenly obvious to any human reader that the text had switched to be machine generated. To this end we set 25% of our prompts to be completely human-written with no generated portion. This ensured that our annotators were more cautious when deciding that a given sentence was machine generated.

The remaining 75% of prompts were sampled randomly from a 20% split of the AI Dungeon training dataset (the other 80% being used to fine-tune GPT-2). The lengths of our prompts are uniformly distributed between 1 and 9 sentences. The sentences were tokenized with the tokenizer described in (Kiss and Strunk, 2006). We also parsed out symbols such as <|startoftext|> and <|endoftext|> that would serve as giveaways to annotators. This was done by re-sampling instead of regular expression replacement to avoid having the context of our sentences switch mid-prompt. Otherwise, all human sampled text is left as-is with any and all grammatical errors or newlines left unchanged.

4.3 Text Generation

Fine-tuning and text generation were done on an NVIDIA Tesla P100 with 16MB of GPU RAM provided by the Google Colab Pro ser-

[1] https://play.aidungeon.io/
[2] https://chooseyourstory.com/

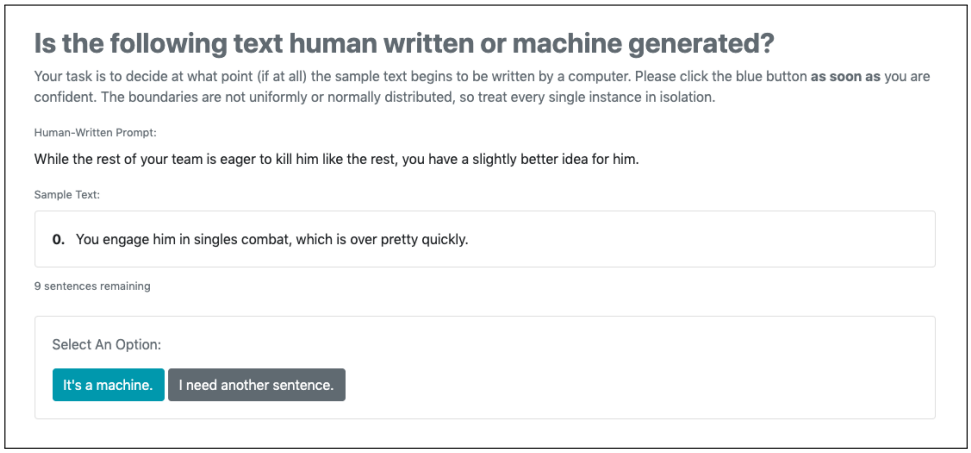

Figure 1: Screenshot of the annotation interface.

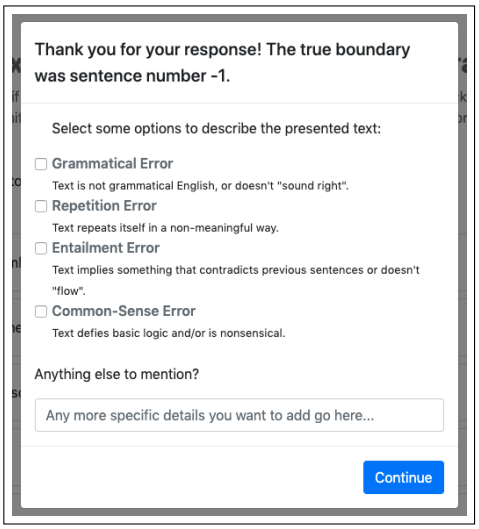

Figure 2: Screenshot of the error taxonomy form.

vice. We used the gpt-2-simple[3] package to fine-tune the GPT-2 (van der Lee et al., 2019) Large 774M parameter model for 1000 steps on 100k lines of training text. We did not tokenize or remove `<|startoftext|>` and `<|endoftext|>` characters from the input to ensure text consistency.

For each generation sample, we prompted our fine-tuned GPT-2 model with only the text from the sampled prompt (and none of the previous context) to ensure that the model was only generating based on context that was also visible to the annotators. The sentence tokens of the prompts were concatenated together by appending newline characters; This is generally consistent with the style of the raw text.

[3] https://github.com/minimaxir/gpt-2-simple

We generated sentences with GPT-2 such that all of our human-machine hybrid text samples had the same 10 sentence length. We chose random sampling and temperature of 1.0 so that the generations accurately reflected patterns present in GPT-2 itself as opposed to artificially minimizing perplexity through other sampling techniques. We leave it to future work to investigate the effects of strategies such as Nucleus Sampling (Holtzman et al., 2019) or top-k random sampling on human perception of generated text.

Sample generation from fine-tuned GPT-2 model:

```
As you close the gap
between you and the elf,
he raises his hand
and kneels before you,
the elf's back and neck
stretching as his
arms around you.
```

5 Evaluation

To evaluate the efficacy of the text generation systems with regards to tricking our human annotators we calculated the accuracy of the annotators in determining which sentence was the boundary between human and machine text as well as the average number of sentences that humans picked before or after the boundary (excluding the 25% of samples that were fully human). We additionally looked at the qualitative categories to identify which errors were the most common and which were the least common.

6 Results

Our experiments consisted of over 150 annotations from 11 different annotators, all of whom had varying experiences with machine learning and computational linguistics. All annotations were collected using our annotation system referenced in Section 3.1, where we maintain the 1:3 ratio between human and machine generated texts.

6.1 Boundary Detection

Our main finding is that only 10% of annotators are able to correctly identify the exact boundary sentence when presented with a machine-generated text (the sample space of this analysis only includes texts with at least one machine-generated sentence). This is compared with a 16% rate of detecting the correct boundary sentence for both machine and human written text (human boundaries being the given prompt).

For all texts which had at least one machine-generated sentence, we observe that 52% of annotators correctly identify the text as being machine generated after the boundary sentence. Additionally, 38% of annotators were incorrect, meaning that they selected a sentence before the actual boundary sentence. In other words, 38% of the time, the annotator thought that a human-written sentence was actually machine generated.

On average, the difference between the selected sentence index and the true sentence boundary was 2.14 sentences. This means that humans remain "tricked" by the system for over two sentences. Future work will involve partitioning our data into the specific generated sentences that tricked humans and the sentences that did not and diagnosing which linguistic features of the text allowed it to fool humans.

6.2 Error Taxonomy

Over 80% of annotations included an error diagnosis. The most common errors listed were Entailment Errors, comprising of 38 of the 150 initial annotations. The next largest error category were the Common-Sense Errors, with 28 distinct annotations. The Repetition Errors and Grammatical Errors only made up 15 and 19 annotations respectively, which is surprising due to the well-known phenomena of repetition in models. Future work will involve allowing users to insert their own error tags, in order to expand the error topology of the project.

7 Discussion

This paper reveals that humans still have difficult with the task of distinguishing between human-written and machine-generated text. However, the initial results of the project presents opportunities to refine the process in order to pursue additional avenues of research. We hope that each addition will allow us to further our research into human classification of machine-generated text.

7.1 Annotator Reliability

As was mentioned in Section 4.2, one of our main goals in this project was to avoid conditioning our human annotators to look for errors in otherwise inconspicuous human text. If we were wholly successful in this regard, the amount of annotators that selected a sentence before the boundary would be 0. While 38% is definitely better than 50%, it still raises doubts about the efficacy of our attempts to caution our annotators against reckless annotating. We must be careful about this 38% and ensure that in future iterations of these experiments, the number is kept as low as possible. We may even have to resort to potentially throwing out certain annotators' data if they are too quick to look for errors where there aren't any.

With that in mind, all results from this project should be considered a lower bound for human accuracy in detection of generated text. In a fully unstructured environment such as a news website or social media post, it is very unlikely that a human will be able to determine that a given passage has become machine generated with over 50% accuracy.

7.2 Error Tags

Although not all of our annotators had a computational linguistics background, future annotators may not understand what the error tags entail. If annotators are not able to correctly categorize errors, it detracts from the overall quality of the error taxonomy. Future work will involve re-naming the current set of tags and including an operational definition with examples.

7.3 Generation Domain

Although the Text Adventure domain prevented annotators from using outside knowledge to assist in discrimination, it sometimes can create noise in the annotation due to the fantasy elements. Some annotations involved annotators mislabelling human-

written sentences as non-sensical due to the necessary fantasy context behind the text. Future work will involve fine-tuning language models on more thematically general domains, such as Wikipedia or Reddit.

7.4 Annotator Signal

Upon making the boundary selection, annotators are immediately presented with the correct boundary sentence. It's possible that this implicitly conditions the type of errors that the annotators are prompted to choose between. Future revisions to the annotation design will be to only present annotator accuracy after completing a set number of annotations (i.e. after they complete the error diagnosis).

7.5 Contextual Inference

One of the problems that plagued our annotators was the fact that the sentences shown which were human-written often-times incorporated names of characters and places which were not mentioned in the prompt. This means that annotators were conditioned to erroneously select certain sentences for entailment errors when, in reality, all information was properly entailed, the prompt just wasn't long enough. We plan to remedy this in the future by only starting to sample from `<|startoftext|>` tokens and never from the middle of human-written stories. This way, if there is an unrelated or unseen name brought up in one of the sentences following the initial sentence, our annotator can be sure that it is the fault of the generation system.

8 Attribution

In the early stages of the project, both Arun and Liam contributed to the experimental design of the process, including the annotation design. This involved reading previous work, and experimenting with large pre-trained language models for avenues of exploration. For the implementation of the project, Arun implemented the annotation system and analyzed the annotation results whereas Liam used large pre-trained language models to generated text and interleave them with human prompts to feed into the system.

Acknowledgments

This work could not be done without the help of Daphne Ippolito and Chris Callison-Burch as well as members from Google Brain team for their feedback on our experimental design. We thank all of our initial annotators for their help evaluating machine written text.

References

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2019. Human and automatic detection of generated text. *arXiv preprint arXiv:1911.00650*.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Comput. Linguist.*, 32(4):485–525.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

João Sedoc, Daphne Ippolito, Arun Kirubarajan, Jai Thirani, Lyle Ungar, and Chris Callison-Burch. 2019. ChatEval: A tool for chatbot evaluation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 60–65, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.