Robust Detection of Computer Generated Text

Haydn Jones

haydn.jones@student.nmt.edu

New Mexico Institute of Mining and Technology

Mentors: Juston Moore, Garrett Kenyon

Background

Recent advances in natural language processing (NLP), based on artificial neural networks, have created powerful tools including grammar correctors, sophisticated machine language translators, and more. Most of these advances use **generative language models**, which efficiently predict the probability of words given their context.

Because they predict words, these models can also be applied to create entire texts. Society must quickly address problems created by the cheap creation of fake multimedia, including automated propaganda, fake news generation, and real-time impersonation.

This work

- **Explores the detectability of current fake text generators**
- **Proposes a baseline attack against detectors proposed in the literature**
- **Develops a machine learning model for adversarially-robust fake text detection based on a sparse coding classifier**

Language Modeling using GPT-2

The current state-of-the-art approach for text generation is derived from the *transformer* attention-based model. There are many variations of the transformer architecture, however, the most well known is OpenAI's GPT-2 (shown right).

- GPT-2 is a probabilistic model; using words 1 through i, it will predict word $i + 1$.
- GPT-2 can assign a probability, p_i, to each word in a text given the previous words.

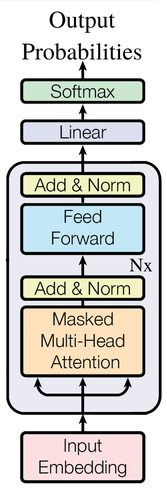
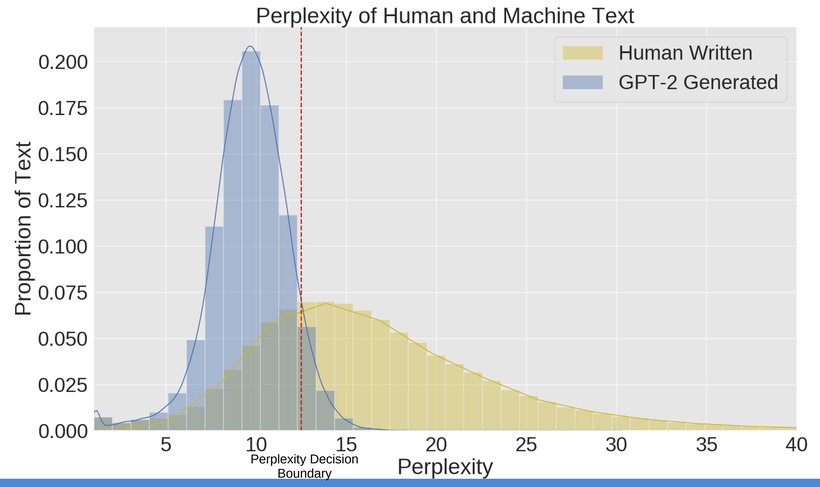


Figure adapted from arXiv:1706.03762

Using the chain rule for probability, we can compute the joint likelihood of all word tokens in a text. The joint probabilities can be used to compute a metric known as **perplexity**, which expresses the number of bits required to encode text in an information theoretic sense:

$$\text{perplexity} = 2^{-\frac{1}{N}\sum_{i=1}^{N} log(p_i)}$$

- Text generated by GPT-2 generally has **lower** perplexity as it models its own text well.
- Human-written text generally has **higher** perplexity since humans use significantly more diverse vocabulary that is not well predicted by GPT-2



Using GPT-2 To Detect Itself

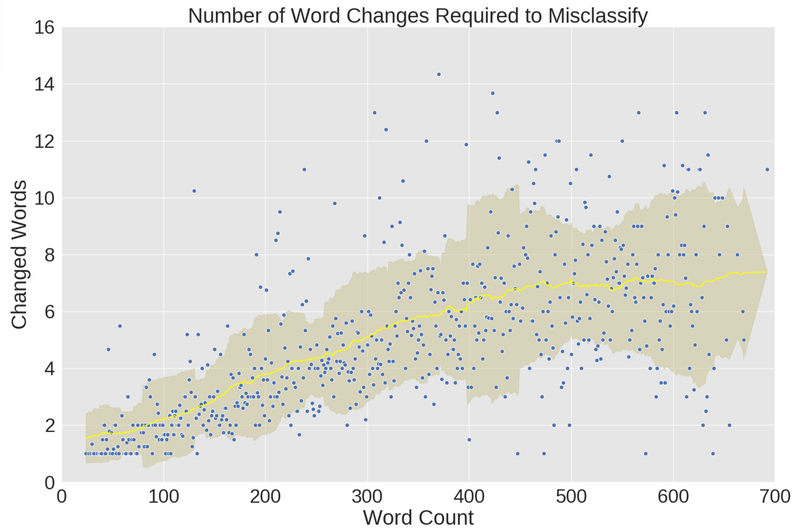
- Due to the difference between word choice in GPT-2 and human text, probability-based metrics are effective discriminators.
- Two classifiers were trained, one on the single value of perplexity, and two others on the probabilities of each word token in a sentence.

Classifier Input	Accuracy
Perplexity	82.0%
Positional Probabilities	97.0%
Histogram of Probabilities	99.3%

- In this work, the Medium (345M parameter) GPT-2 model is being used to classify the Extra Large and unreleased (1542M parameter) GPT-2 model.

Adversarial Greedy Algorithm

- We seek to replace words with close synonyms based on the Word2Vec NLP model, which maps words to a vector space where similar words are close together.
  - For example, a Word2Vec model will likely group "ran", "sprinted", and "jogged" close together in vector space.
- We developed a novel greedy algorithms that iteratively replaces words in text created by GPT-2 with poorly-predicted synonyms until a perplexity-based classifier fails.
  - For each word in the text, we look at N alternative words, and choose the one that raises perplexity the most and repeat until misclassification.



On average, one word per hundred in the text must be changed to trigger a misclassification on a perplexity-based classifier.

Example of an Adversarial Example:

- "We've teamed up with the folks over at Jamey Stegmaier to bring you an exclusive preview of a couple new screenshots. One gives you a ~~clear~~ abundantly clear view into the cockpit and the other looks at the inside of the hull."

Takeaways

- **Current text generators are surprisingly effective, but they have fundamental flows which allow easy detection.**
- **Detection of text generated by GPT-2 is easy, however, it is possible to deceive a simple detector by changing a small number of words.**
- **In an adversarial environment, using GPT-2 to detect itself is not robust.**
- **Sparse coding offers a robust classification method for fake text.**

The Sparse Coding Approach

Sparse coding is an unsupervised machine learning algorithm that attempts to find efficient representations of data by optimizing the following equation over ϕ for the entire dataset, and over a for each input text x_i:

$$\min_a || \overbrace{x_i - \phi a_i}^{\text{Residual}} ||_2^2 + \underbrace{\lambda ||a_i||_p}_{\text{Sparsity Penalty}}$$

- Φ is a collection of **features** called a dictionary.
- a is **sparse code** for a particular text representing a linear combination of features in Φ. Each coefficient is called an **activation**.
- λ is a sparsity hyper-parameter for the sparse coding algorithm.

- By replacing words in a text with their Word2Vec representation, it is possible to build a sparse coding model to efficiently encode text.
- A classifier trained on the average activations of the features encoding a piece of text has a classification accuracy as follows:

Classifier	Accuracy
Average Activations	94.4%

- Despite having a lower accuracy than the probability-based classifier, sparse coding is more robust to adversarial attacks than the probabilistic models.
- This is an initial result. More work can be done to improve this classification score
- **Results of perplexity and sparse coding on adversarial inputs:**

Model	Perplexity	Sparse Coding
Base Accuracy	100.0%	75.0%
Correct → Incorrect	100.0%	2.6%
Incorrect → Correct	0.0%	2.6%
Adversarial Accuracy	0.0%	75.0%

Future Directions

- **Classification of other Models:** More performant generative models are being designed, such as Grover. Will sparse coding continue to perform well, and will it remain robust to adversarial examples?
- **Sparse Coding:** This is the first use of sparse coding on text, does it perform well on other NLP tasks?