Reddit Language Modeling[[1]](#footnote-0)

Matthew Scharf and Alex Seidel

Spring 2021 Final Project

CIS 522: Deep Learning

University of Pennsylvania

1. Introduction

[Reddit](https://www.reddit.com/) is a large website containing many online forums where people gather to socialize, share memes, give advice, plan businesses, and so much more. Each of these forums are called subreddits and each subreddit centers around a topic or theme (e.g. Game of Thrones). Reddit has many users and new posts each day, at a scale where organizing and monitoring text must be done automatically if it is to be done at all.

As we all know from the news, monitoring online forums like Reddit is crucial for guaranteeing the prevention of large-scale misinformation campaigns, election manipulation, terrorist threat identification, and much more. From a slightly less drastic perspective, being able to monitor and evaluate text would allow for strategies like spam flagging and recommendations that make Reddit more enjoyable and useful. For example, perhaps a question about story writing on a movie subreddit would be better suited for a script development subreddit: that subreddit could then be recommended to the user.

As we know from CIS 522, such automated systems can be greatly aided by the use of machine learning to approximate language distributions. To that end, the objective of our project is to build language models which represent the distributional characteristics of reddit text.

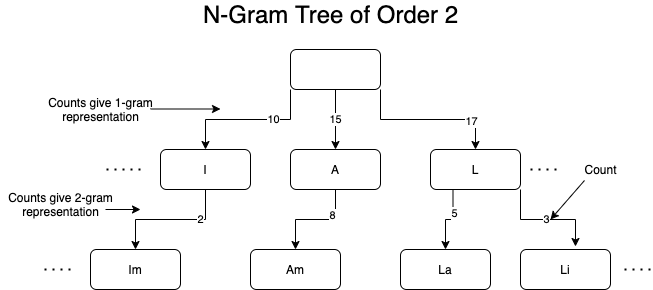
Specifically, we hope to solve the following three tasks with machine learning:

* **Task 1:** Generate representative text for each of several large subreddit categories
* **Task 2:** Classify a new text into one of these subreddit categories
* **Task 3:** Distinguish real text from fake text (e.g. text generated from Task 1.)

Our data set comes from kaggle.com and can be found at <https://www.kaggle.com/mswarbrickjones/reddit-selfposts>. The dataset provides ~1 million posts from 3394 subreddits. These subreddits are further grouped into 38 categories. We randomly choose 5 categories on which to focus:

1. Tv\_show: 68k posts
2. Electronics: 51k posts
3. Writing&stories: 22k posts
4. Autos: 20k posts
5. Hardware&tools: 14k posts

2. Non-neural Baseline: n-gram Tree (code found [here](https://colab.research.google.com/drive/1cKaSaGovh4We6GgYj7mn0K-2lOkwUUTS?usp=sharing))

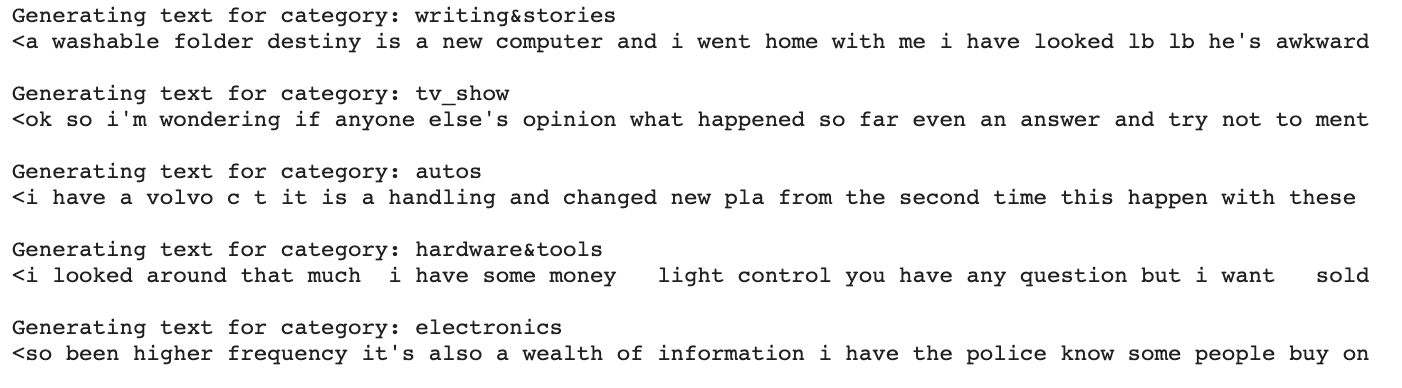
We first train 5 character-level n-gram trees on each of the 5 subreddit categories as a non-neural generative language model (Task 1). We can also use these trees for classification (Task 2). N-gram trees maintain counts of transitions from a history of length 0,..,k-1 to the next character where k is the order of the tree. We can then normalize these counts over all counts stemming from the same source to obtain a probability distribution. We can see from the following image how this corresponds to maintaining n-grams of size 1-k. 

We use an absolute discounting technique (with a discount=.9 as the n-gram history shrinks by one character) to average the transition distributions over all history sizes. This improves robustness, stability, and generalization over a pure n-gram approach because the smaller n-grams provide a nice simplifying bias.

For each subreddit category, we use all data available, a tree order equal to 10, and train for one epoch.

2.2. Text Generation

After we have the above tree and the resultant smoothed probability distribution over the next character given a history, text generation is as simple as repeatedly sampling from this distribution. The following demonstrates what that looks like for the five categories:

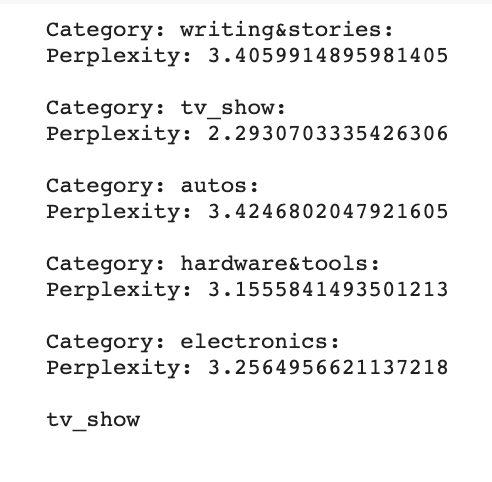


The text seems to have phrases relating to the subreddit class (e.g. “higher frequency” in electronics, “volvo” in autos, and destiny in “writing&stories”) but the model seems to struggle over longer texts to form coherent sentences. This makes sense as the memory of the model is only 10 characters long.

2.3. Text classification

We can use the n-gram trees to classify text into one of the five subreddit categories using the following strategy: calculate the perplexity per character of the model on the given text and choose the category corresponding to the model with the lowest perplexity. Below, we can see an example of this approach used on text from the “Tv\_show” category:

**Text: “I just finished Scandal and I’m looking for a new show.”**



We see that this text is properly classified into the tv\_show category as that model results in the lowest perplexity.

3. Basic neural: GPT2-Based Reddit Post Generation (code found [here](https://colab.research.google.com/drive/1PgPNoN83Dh8bbqCJS1jFc-1UgII9Ms9z?usp=sharing))

Here, we fine-tune GPT2-Based neural networks to build generative models for each of the 5 Reddit categories (Task 1). We use Huggingface to do this (in fact, we will use [Huggingface](https://huggingface.co/) for all of our neural language models) and use the “[distilgpt2](https://huggingface.co/distilgpt2)” model as our base model. Distilgpt2 is a smaller (fits easily on google colab) approximation of [GPT2: a large pretrained generative language model](https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf) consisting of a series of stacked decoders. For each category, we used all data available and trained for 3 epochs. After training, the models produced the following generated text:

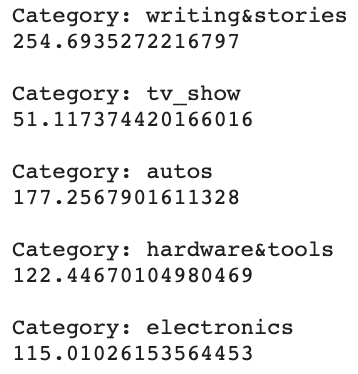
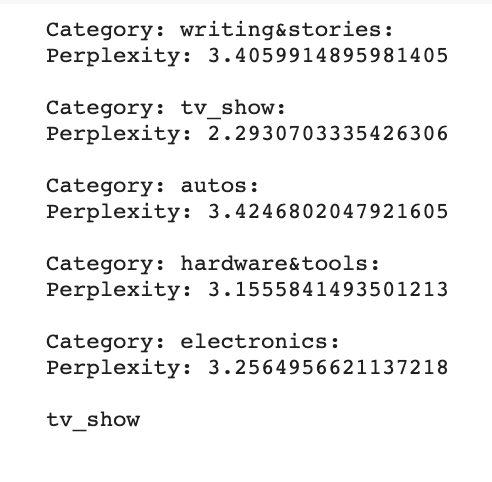
* **Writing&stories:** What I think ive read so far is a fantastic description of the project i'm really looking forward to doing lb lb i'm looking for a writer to fill the role of the girl lb
* **Tv\_show:** What I think ive heard has been a relatively new phenomenon i've been following for awhile and i feel like the rise in frequency of stories on the web and in my
* **Autos:** What I think ive been driving a lot ever since the 's was an excellent toyota for years and a few times i had a hard time getting past the clutches and it seemed obvious
* **Hardware&tools:** What I think ive just made a mistake i'm planning on replacing the scooter i've owned since i was a kid but im looking for something that's both fun and fun
* **Electronics:** What I think ive tried and tried the zx gopro and the xy and ive tried them as a side note all my favorite features were the motion detection for the xy and the motion

We see that each text exhibits characteristics from the category on which it was trained. Furthermore, in comparing this generated text to the text generated by the n-gram trees, we see that this text is much more fluent and coherent both grammatically and semantically.

Additionally, just like with the generative n-gram models, we can calculate average perplexity and perform classification by choosing the subreddit category associated with the smallest perplexity. We apply this method to the same text as above for comparison:

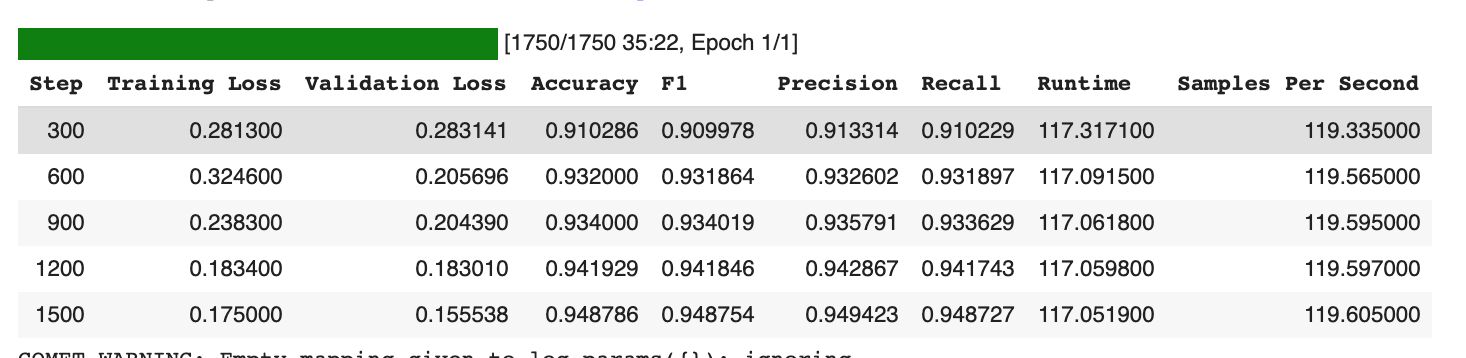
**Text: “I just finished Scandal and I’m looking for a new show.”**

**N-gram Trees GPT2**

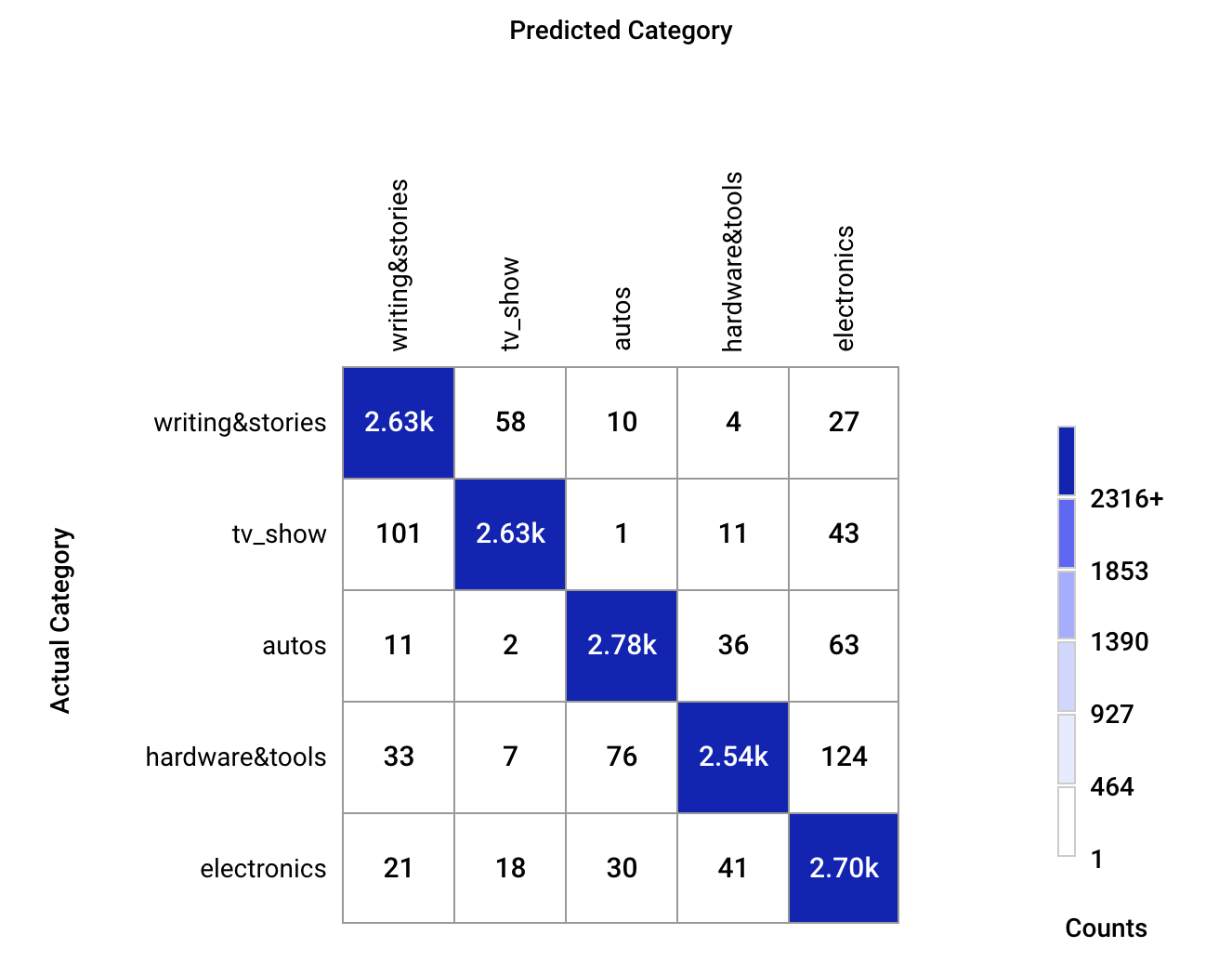


We see again that our generative models are accurately able to classify the text as belonging to the tv\_show category. It seems as though the GPT2 models are even more confident of this classification due to the large discrepancy between the tv\_show perplexity and the others. Furthermore, you might notice that the perplexities are higher across the board for GPT2. This is because it was pre trained on a huge corpus and so therefore takes into account a much broader range of possibilities than our model which was only trained on in-distribution text.

4. Basic neural: BERT-Based Classification (code found [here](https://colab.research.google.com/drive/1te7Pom7qqc13zzU3Wh0pzKK0MzjIFHdv?usp=sharing))

We again leverage Huggingface to train a model which can classify a given text into one of the five subreddit categories (Task 2 above). This time we use the “[distilbert-base-uncased](https://huggingface.co/distilbert-base-uncased)” model as our starting model. Distilbert is a smaller approximation of [BERT: a large pretrained discriminative language model](https://arxiv.org/abs/1810.04805) consisting of many stacked encoders. To train this model, we sample 14k datapoints from each dataset for a total of 70k posts. We use an 80/20 train-test split and train for a single epoch:

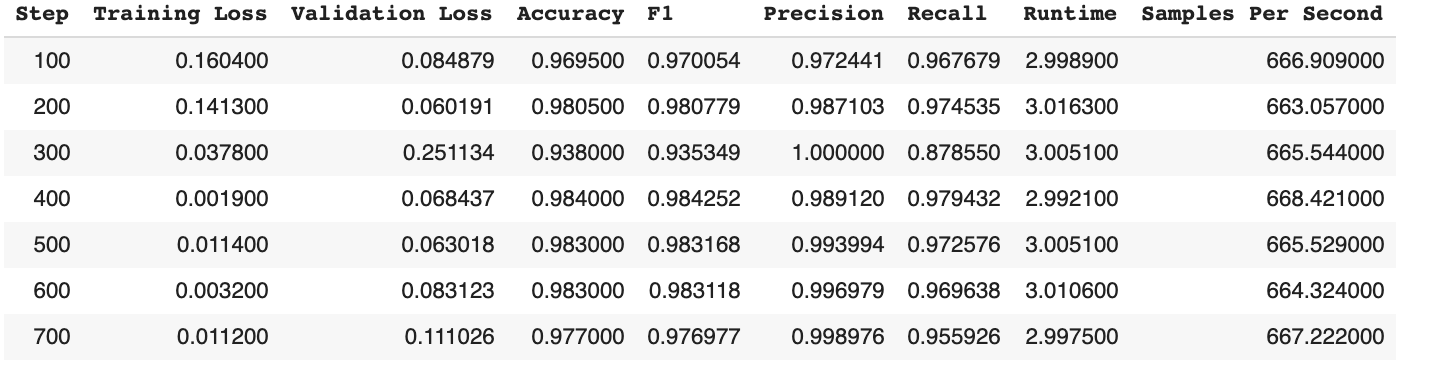
We see that the training is successful and that we are able to achieve a fairly high ~95% accuracy and F1 score on our test data. By looking at the confusion matrix, we get a better idea of where the mistakes are occurring:



We see that the model confuses “writing&stories” with “tv\_show” as well as “eletronics” with “hardware&tools”. This makes sense as we know that in both instances the subreddit categories are semantically similar.

5. Advanced neural: Discrimination of Real and Generated Posts (code found [here](https://colab.research.google.com/drive/1HcrfomE6UMQQrV3W3XlZIwe3rD-jkBG4?usp=sharing))

Here, we once again use the huggingface “[distilbert-base-uncased](https://huggingface.co/distilbert-base-uncased)” model as the base model for a binary language classifier that distinguishes real text from generated text (Task 3). We train this model on 1000 real posts sampled from each category, as well as 1000 posts generated by the GPT2 generator trained for each category, to give a total of 5000 real and 5000 generated posts. The model is then trained to classify these posts as real or fake, irrespective of the subreddit category. The discriminator is then trained on this data for three epochs using a 80/20 train/test split:



The classifier learns very quickly, achieving a test accuracy and F1 score of 98% only after 100 steps of training.

6. Conclusions

Ultimately, using the kaggle reddit data set, we were able to build several different useful models. First, we used an n-gram based approach to create generative language models for each subreddit category and classification of posts by choosing the category for which it has the lowest perplexity. Then, we used GPT2-based generative models for each subreddit category and a BERT-based classifier for posts from the five categories. Finally, we trained a binary BERT-based language classifier to discern posts generated by the GPT2 models from real reddit posts in the five categories.

While the neural models had noticeably better performance than the n-gram based approach at text generation and classification, another advantage of the neural models is their much greater speed both in training and using. The space and time taken by the n-gram model grows quickly with the order of the tree. A 250-word post that takes five minutes or so to generate using the n-gram tree takes only a few seconds using a trained GPT2 generator.

Despite the success of our models, it is worth noting that the generated language discriminator is likely limited in its ability to generalize to new contexts. While the model was trained on data from 5 different subreddit categories, it was only trained on our kaggle reddit data and text generated from GPT2 generators. That being said, the GPT series are state-of-the-art for language generation, and reddit posts are fairly representative of text found across the web. As a result, we anticipate that the generated language discriminator would remain useful for a variety of tasks on Reddit and possibly even some internet-text-related tasks outside of Reddit.

Lastly, it is important to acknowledge the issues of bias and privacy that arise from the use of such large pretrained language models. GPT2 and BERT both have a fair amount of inherent bias (see [here](https://huggingface.co/bert-base-uncased#limitations-and-bias) for example) due to the large amount of bias present in their training corpi. As a result, we need to be careful when using these models and avoid using them to make decisions which perpetuate biases and discriminaton. In regards to privacy, by using this text data to train our model, we are likely using it in a way that the authors did not anticipate. When collecting data, it is important to clearly notify users regarding how their data will be used and allow them to opt out of collection ahead of time.

1. Project code can be found [here](https://drive.google.com/drive/folders/1WHLh0ksOhqkBHk6hXKpywPjlPFkLTWDP?usp=sharing) [↑](#footnote-ref-0)