Generating Text from Pretrained LMs

Group (Comp E Kids): Hamza Khan, Zeb Zimmer, Ben Moorlach
Github Repository (see Homework3 folder): https://github.com/ZebZimmer/NLP

Task 1: Decoding Algorithm Implementation

For all tasks in the assignment, the *gpt2* pretrained model was used.
4 decoding algorithms were implemented when calling *model.generate(...)*.
The same prompt of "It might be possible to" was used with all generation tasks in Task 1.
Algorithm parameters and generated output metrics are shown below:

1. Beam Search Parameters: *num_beams=3, early_stopping=True, max_length=30*
2. Greedy Search Parameters: *do_sample=False, max_length=30*
3. Top-K Sampling Parameters: *top_k=50, do_sample=True, max_length=30*
4. Top-P Sampling Parameters: *top_p=0.8, top_k=0, do_sample=True, max_length=30*

The max_length was designed to be the same for all of the methods as larger values became more incomprehensible. The number of beams (3) was chosen because larger values ran longer and also increasing it led to memory overflow errors. The top K was chosen as larger values led to memory issues again but the top P value was chosen arbitrarily.

The value of the logits that were used in calculating perplexity and likelihood were found using code from this hugging face tutorial/documentation page. The scale of perplexity feels wrong and should likely be taken only in the context and relation of this table specifically. The code and outputs of Task 1 can be seen with the rest of our work at this github repo which may have slightly different values than reported below due to hyperparameter tinkering and whatnot.

Table 1: Initial performance of the output from the 4 decoding algorithms

Algorithm	Perplexity	Likelihood
Greedy Search	5.51	0.792
Beam Search (3 beams)	4.37	-9.34
Top-K Sampling	9.83	13.4
Top-P Sampling	13.64	2.02

Task 2: Decoding for Extrinsic Evaluation

For this task, we elected to use the *cnn_dailymail* dataset from huggingface seen here. The dataset is designed to train models for summarization tasks. From huggingface's explanation of the dataset, each entry contains an *article* field which is a string containing the body of a news article, and a *highlights* field, a string containing the highlight of the article as written by the article author.

The goal of using this dataset was to summarize each article using the different decoding algorithms within the *model.generate(...)* function. The generated summaries were then compared with the ground truth value within the *highlights* field. The Nx5 spreadsheet is linked alongside this document since it is too large to show here. Here's the link to view it in Google Sheets.

Task 3.1 Automatic Evaluation

Table 2: Decoding algorithm performance with various metrics

Algorithm	Meteor AVG	BERT (Precision) AVG	Rouge1 AVG	Perplexity AVG
Greedy	37.12	50.54	35.61	748131
Beam	38.33	49.82	35.66	758466
Top-K	38.64	49.61	35.64	556414
Top-P	38.14	49.63	35.14	1510695

The average perplexity of the ground truth data was 3327372.

We chose to do 4 metrics for two main reasons: the first being that we want the extra credit points, the second is that we were unhappy with the similarity of the scores. Besides perplexity the scores are wildly similar for each of the metrics. Our thinking for why is concerned with the fact that we used a very finely tuned model (facebook/bert-large-cnn) and the dataset we tested on was the dataset the model was fine tuned on. The reported Rouge1 score for the model was ~42 which puts us close with only 50 of the data sets. We assume if we dialed in the hyperparameters we would converge on the ~42 score. Ultimately the 4 decoder methods we used did not vary enough to stray the metrics away from the model's general scores for the given metric. It would have been more interesting and insightful to use a more general model as therefore the different decoder methods would have had more chances to vary. Therefore, the scores are more indicative of what the model would score with a general set of hyperparameters and the efficacy of each decoder algorithm should be understood with respect to the themselves. Meaning, simply, the take-aways from our work should be along the lines of: theGreedy decoder had the best BERT of the bunch but BEAM had the best summarizations because it had the highest ROUGE1.

The hyperparameters we chose were the same as the ones used in task 1 except for a few notable differences. The first being that max_length = 100 which is due to the model for task

2 required max_length to be greater than 56. We chose 100 because it seemed fitting. The second difference is that top_k and top_p were decreased to decrease runtime and prevent memory issues.

The four metrics came from Hugging Face's evaluate-metric function. Meteor is a good metric to use because it is based on unigram matching between machine-produced translation and human-produced reference translations. We also used the precision score from BERTScore which works well for measuring language generation tasks. Then we chose to use the ROUGE1 score which is again unigram based scoring so that it could be easily compared to the Meteor average. ROUGE1 is a good metric to use because it's designed for evaluating automatic summarization which is exactly what we were doing. Finally we used Perplexity because we wanted to do another model-based metric like BERT for the extra credit. We unfortunately are unsure of why perplexity is so great here and why it's so much less than the perplexity of the ground truth. One line of thinking is that the Perplexity metric from Hugging Face did not normalize to the length of the text. It is rather obvious when reading the outputs that the model's summarizations were less "perplexing" than the ground truth but the scale of the difference between perplexity values is, well, perplexing.

Task 3.2 Human Exploration

A human annotated spreadsheet of 20 generated article summaries is shown here. The model's generated summaries were ranked using a Likert (1-5) scale. The four decoding algorithms greedy search, beam search, top-k, and top-p were compared.

We evaluated each generated summary using two metrics: coherence and factuality. Coherence was an easy choice for a metric. We wanted to see if the generated outputs actually made sense. Incoherent outputs are not useful to us in regards to article summary. The second metric chosen was factuality. When summarizing news articles, we want to be sure that the model does not generate incorrect or fake news. Misinformation is a problem, and we don't want our language models to contribute to it.

In general, the generated outputs had high coherence scores across the board. In this case, factuality was sometimes difficult to evaluate. However, it was obvious when one of the outputs was factually incorrect. Above, in section **3.1**, we saw how our automatic evaluation metrics didn't show much variation in performance between the different decoding algorithms. These performance metrics can only capture so much. As a result, human evaluation allows us to see which generated output made the most sense to us as humans.

The average values for each decoding algorithm across both metrics is shown in the table below: As you can see, Beam Search was ranked with the highest average coherence, and it also had the highest average factuality. However, all of the search algorithms performed pretty well and had similar performance overall. The reasoning behind this is the same reason as in section **3.1**, describing how our model was actually trained with some of our dataset.

There are patterns to be seen between the blind-human evaluation and the automatic evaluation. Beam Search had the highest rated coherence and factuality and also the greatest

ROUGE1 score which makes sense as ROUGE1 was made to evaluate specifically automatic summarization. Top-P Sampling had the worst, or tied for the worst, scores and also had the highest Perplexity score of the group as well as the lowest ROUGE1.

Table 3: Human annotated ratings for output from each decoding algorithm

Algorithm	Average Coherence (out of 5)	Average Factuality (out of 5)
Greedy Search	4.52	4.20
Beam Search	4.70	4.28
Top-K Sampling	4.57	4.25
Top-P Sampling	4.33	4.20