Computer generating movie reviews. With GPT-2, you can too!

Ole Jonas Liahagen and Martin Johannes Nilsen
NTNU, Computer Science, 2022
[olejlia, martijni]@stud.ntnu.no

Abstract

Text generation, also known in the literature as Natural Language Generation, targets the creation of text intending to be almost indistinguishable from human-written text. Early implementations saw the use of Markov processes or deep generative models such as RNN or LSTMs. Although being able to generate creative texts, the latter approach suffers from long training times. This has led to the rise of transformers in the field, as this attention-based architecture yields great improvements in training time, in addition to improved model accuracy. We con- tribute with a dataset consisting of approximately 70.000 movie- rating and review pairs in the GPT-2 format, in addition to a fine-tuned GPT-2 model. Using this model, one could generate somewhat creative reviews of a non-existing movie, prompted with a rating on a scale of 1-10.

1 Introduction

The development of text generation models began decades ago, although in quite a different format from today's architecture of choice. Early text generation models were trained using Markov chains. Each word would in this case be represented as a state of the chain. For calculating the probability of a word being the next one, the number of consecutive occurrences of both the previous and the current word in the training text would be emphasized. Following the use of Markov chains, Recurrent Neural Networks became the successor for generating text. Long Short-Term Memory models were capable of retaining a greater context of the text introduced, as it contains a memory of former words. This kept being the favored choice until the release of the Transformer architecture in 2017, which will be further elaborated on in Section 3.

The field of review-related applications of NLP is not a new phenomenon. Being able to extract the most important opinions or sentiment of any product can have immense value business-wise. Some common implementations include the use of product reviews[1] and restaurant reviews[2], but especially movie reviews seem to have gained traction. The most commonly used dataset, called the *IMDB dataset*, was presented by Maas et al. (2011) and has at the time of writing 1168 papers having acknowledged using it[3]. The dataset includes 50.000 user reviews with an associated rating intended for sentiment analysis. No more than 30 reviews are included per movie, and only highly polarizing reviews are considered. Although the focus on NLP and movie reviews tend, in our perception at least, to be related to extracting information or performing analyses of different sorts, some implementations of text generation can also be found. From story plot generation presented by Gervás et al. (2004), to persona generation (Bamman et al., 2013) and product review generation (Dong et al., 2017), there seems to be an increased interest in text generation in recent times. This could arguably be due to the recent work of OpenAI, releasing their Generative Pre-trained Transformer models intended for improved language understanding. These models have taken the natural language community by storm, showing promising results in generating human-like text. Regarding the generation of movie reviews specifically, it does not exist a relevant paper to the best of our knowledge, although there are multiple online articles on the matter. This is the

[1]Popularly cited papers include Hu and Liu (2004), Popescu and Etzioni (2007) and Kang and McAuley (2018)
[2]Popularly cited papers include Kang et al. (2012) and Pontiki et al. (2014)
[3]According to `https://paperswithcode.com/dataset/imdb-movie-reviews` (Meta AI Research)

motivation behind this paper, trying to fine-tune a GPT-2 model on movie reviews, for generating reviews with a prompted rating.

2 Background

This section presents background work relevant to fine-tuning a state-of-the-art model for text generation. More specifically, the topics to be covered include transfer learning, which is the basis for model fine-tuning, and the multiple ways of deciding the next word in a sentence, for generating the most human-like texts. Finally, the `aitextgen` module will be explained, being the framework of choice for this paper.

2.1 Transfer Learning

The concept of transfer learning is a popular approach in machine learning, which deals with using knowledge of how to solve a type of problem and use it to solve a new, related one. Training recent state-of-the-art models are computationally heavy and thus costly, which makes the use of transfer learning even more lucrative. With larger companies using their massive amounts of computational power in training machine learning models, a trend has been in recent years to make pre-trained general models publically available, which can be further fine-tuned for solving more specific tasks.

While being used in many forms of machine learning, the area of text and natural language is no exception. With the general models being trained on massive amounts of data, the models gain a good grasp of how language works in general. Further on, the language model can be trained on a dataset of desired output format, specializing the model to the task.

2.2 Decoding method

For being able to generate meaningful text, the probability distribution of the most-suited next word, generated by the language model, needs to be decoded. Several decoding methods have been proposed, including greedy search, beam search, and sampling.

Greedy search always selects the word with the highest probability. One major drawback of this approach is the fact that high-probability words often hide behind low-probability words, and greedily selecting the route based on the next best word is not always the best in the long run. Beam search addresses this problem by keeping the most likely routes (beams) at each time step, and eventually choosing the highest probability by comparing the outcomes of each of them. Take for example a beam search with 2 beams, the continuations to be considered include all the combinations of 2 consecutive words, selecting the one with the highest probability of them. As it turns out that our language does not necessarily follow a high probability next word distribution (Holtzman et al., 2019), other decoding methods have been developed to address these issues.

Sampling is the process of randomly picking the next word according to the probability distribution extracted from the language model. Random means not deterministic and could therefore fit the task of mimicking a human. The first common approach of sampling is Top-k sampling, filtering the k most likely next words, before sampling on these k words only based on their likelihood scores. The use of Top-k sampling often leads to the generation of more human-sounding text than the formerly mentioned decoding methods. One concern though is that it does not, as k is fixed, dynamically adapt the number of forts that are filtered. Very unlikely words may therefore be selected among these k words if the next word probability distribution is very steep.

This led to the proposal of nucleus sampling (Holtzman et al., 2019), also known as Top-p sampling. Top-p sampling chooses from the smallest possible set of words whose cumulative probability exceeds the probability of p. This way, the number of words in the set can dynamically increase and decrease according to the next word probability distribution. Top-p is usually set to a high value, i.e. 0.75[4], with the purpose of limiting the long tail of low-probability tokens that may be sampled.

Both Top-k and Top-p sampling is used in modern implementations, and worth mentioning is the option of implementing both at the same time, where the Top-p sampling acts after the Top-k sampling.

[4]`https://docs.cohere.ai/docs/controlling-generation-with-top-k-top-p`

2.3 aitextgen

In the early stages of the project, the module `gpt-2-simple`[5] was utilized as it provides wrapper functions for loading, fine-tuning, and inferencing using a pre-trained GPT model and was easily accessible from the Python Package Index[6]. This was later on replaced with what was described as its successor, the more recent module called `aitextgen`[7]. This module provides everything the former one did and more, with higher efficiency, an expanded list of pre-trained models and tuneable hyperparameters, and the ability to create compressed tokenized datasets for model training.

3 Related Work

A lot of related work has been done before us, enabling the fine-tuning of a generative pre-trained language model, with the intention of creating reviews from prompted ratings. The purpose of this section is to highlight a selection of the most important relevant work that lays the foundation of the field, being the transformer architecture and the Generative Pre-trained Transformer (GPT) model.

3.1 The transformer architecture

The transformer architecture, proposed by Vaswani et al. (2017), was intended to improve the language model's ability to handle long sequences of text. With the help of three key features, respectively positional encoding, attention, and self-attention, the transformer architecture has been wildly popular in NLP implementations, with the original paper being at the time of writing cited by over 50.000 papers[8].

Following the original paper, Liu et al. (2018) proposed another arrangement of the transformer block, intended for text generation use. Throwing away the transformer encoder, the paper suggests a language model made up of a stack of six transformer decoder blocks only (Figure 1).

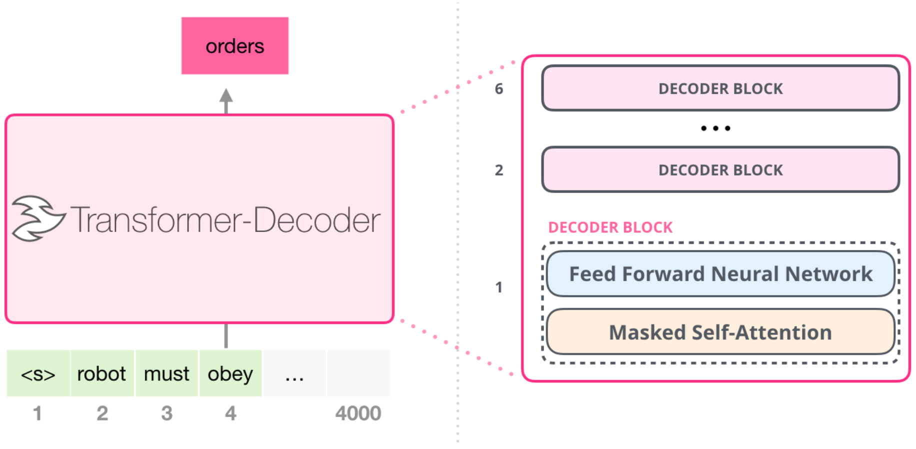


Figure 1: The transformer-decoder arrangement[9]

3.2 Generative Pre-trained Transformer

The GPT, or Generative Pre-trained Transformer model, is arguably the most frequently used model for text generation in recent times. The original GPT model, presented by Radford et al. (2018) in OpenAI, had a decoder-only transformer structure, using the decoder-blocks described in Figure 1. Following the success of the original model, Radford et al. (2019) propose the GPT-2 model, having 1.5 billion parameters against the 117 million parameters in the original one. The model was trained on 40Gb of

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

[6]`https://pypi.org/project/pip/`

[7]`https://github.com/minimaxir/aitextgen`

[8]`https://scholar.google.com/scholar?hl=no&as_sdt=0\%2C5&q=attention+is+all+you+need&btnG=&oq=attention`

[9]Illustration from Alammar (2018a), published under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

web text, the result of the authors scraping Reddit, pulling data from highly upvoted articles' outbound links. The resulting dataset, WebText, combined textual data from over 8 million documents. This was a huge improvement over the dataset used for training the original GPT model, the BooksCorpus dataset. The GPT-2 model is the model of choice for this paper, although not the largest of the available parameter configurations. This will be further elaborated in Section 4.

4 Architecture

In this section, the pipeline architecture from dataset expansion, preparation, and tokenization, to feeding the data for fine-tuning, will be elaborated. Furthermore, the text generation will be described, that is, inference using the fine-tuned model.

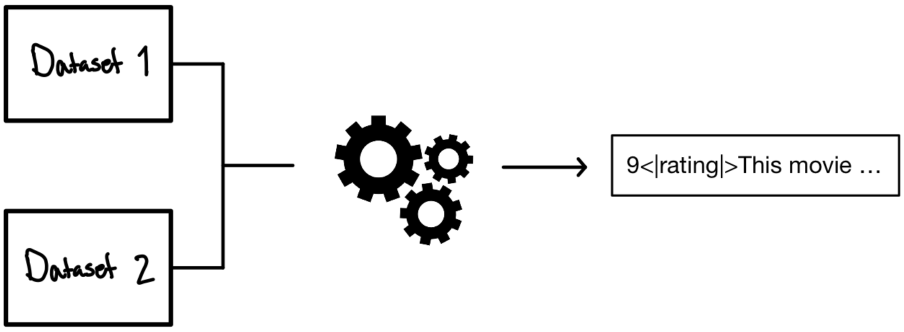


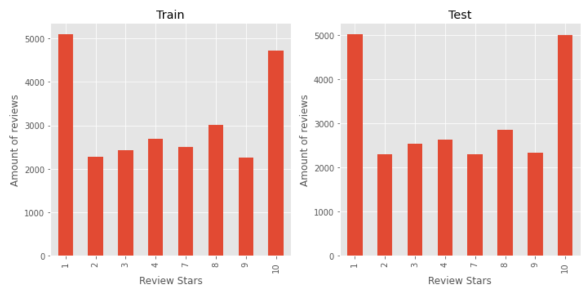
Figure 2: Dataset creation pipeline

4.1 Dataset creation

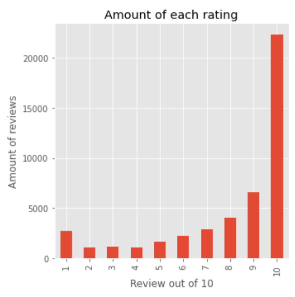
As mentioned in Section 1, the IMDB dataset by Maas et al. (2011) has been a wildly popular choice for movie review-related tasks in recent times. Therefore, the dataset was seen as an obvious choice in this paper as well. After exploring the dataset, which is initially split into a training and testing portion (Figure 3a), the effects of the imbalance were questioned. In addition, the original dataset does not include reviews with a rating of 5 or 6, as it wants to be polarizing for the sentiment analysis task it is intended for. This was the reason for introducing a second dataset, a dataset[10] fetched from the data science platform Kaggle. This dataset includes just below 50.000 user reviews from 10 popular movies. Figure 3 shows the distribution of rating counts. The initial dataset of Maas et al. (2011) seems to have approximately double the amount of one- and ten-star reviews (Figure 3a). The rest of the counts for that particular dataset seems to be quite balanced. Combining the intended two datasets of 25.000 training and 25.000 test samples, the imbalance may affect the outcome. Thus, the second dataset was introduced (Figure 3b). For combining these, the one- and ten-star reviews were left out, and one can observe in Figure 3c that the expansion also includes a smaller portion of five and six-star reviews.

For the GPT-2 model to understand the contextual meaning of the data provided, it will perform a process called tokenization. In this process, blocks of text are separated into units, being either words, characters, or subwords, thus making up the model's vocabulary. GPT-2 specifically, utilizes a Byte Pair Encoding approach. Here, co-occurring subword pairs are merged, forming the model tokens. The use of a subword-based tokenizer does create the need for training one, but have the major advantage of being better at dealing with out-of-vocabulary words (Sennrich et al., 2015), while also reducing the number of model parameters. As the vocabulary used in movie reviews is unlikely to contain a vast amount of terms differing from what the GPT-2 original BPE tokenizer was trained on, being 40GB of web text as mentioned in Section 3, there is no reason to train a BPE tokenizer from scratch. The model will then have to figure out what to do with the new tokens, if there are any, by updating the embedding layer. Nevertheless, this should be feasible.

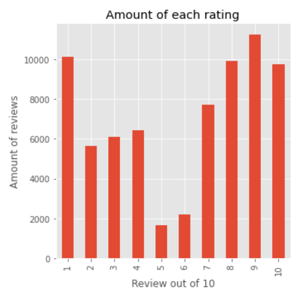
[10]`https://www.kaggle.com/datasets/sadmadlad/imdb-user-reviews`



(a) Original dataset



(b) Kaggle dataset



(c) Extended dataset

Figure 3: The dataset, consisting of approximately 70.000 movie user reviews

After having merged the two datasets, the data was then ready for being processed into a set of model-understandable prompts, as illustrated in Figure 2. The `aitextgen`-framework, as mentioned in Section 2, provides a class called `TokenDataset` for compressing and tokenizing a dataset. The use of this package will have an impact on model training, saving crucial GPU RAM which can be the difference in being able to train the 124M vs. 355M hyperparameter version in Google Colaboratory VM.

For the creation of a tokenized dataset, multiple steps were taken. The first of these was to clean the data. After manual inspection, a set of textual elements were decided to be important in the text preprocessing. All reviews were sequentially run through a pipeline that removed emojis, Html tags, and whitespace, in addition to both unnecessary and/or repeated punctuation. A prompt was then made for each review, containing the rating, a token in between, and the review itself.

”9<|rating|>This movie was so close to being the best in the franchise! ...”

As seen in the example prompt above, the token separating the rating and text has the format of `<|rating|>`. This is inspired by the default end-of-file-token used by GPT, being `<|endoftext|>`. Using the `tokenDataset`-class provided by `aitextgen`, the end-of-file-token (or end-of-sentence-token, which is the same one) is appended line by line when creating the tokenized dataset. This dataset is then further compressed, and ready to be passed on to the model trainer.

4.2 Model fine-tuning and inference

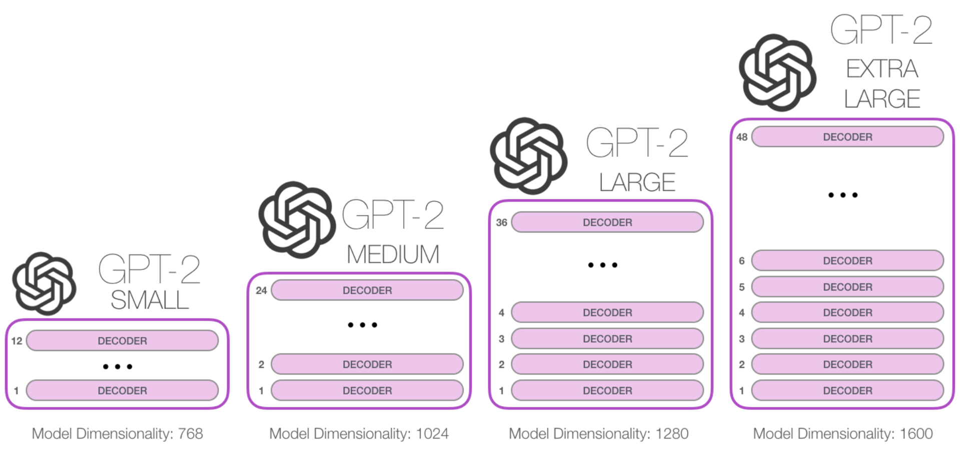

GPT-2 SMALL
GPT-2 MEDIUM
GPT-2 LARGE
GPT-2 EXTRA LARGE
DECODER
Model Dimensionality: 768
Model Dimensionality: 1024
Model Dimensionality: 1280
Model Dimensionality: 1600

Figure 4: The selection of GPT-2 models[11]

As illustrated in Figure 4, OpenAI provides four different sizes of the GPT-2 model architecture. Before the addition of the `TokenDataset`, only the 124M hyperparameter version of the model was

[11]Illustration from Alammar (2018b), published under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

trainable in Google Colaboratory VM without running out of GPU memory. In the contributed notebooks[12], the 355M hyperparameter option was the final choice, being the largest model that the environment could handle and was seen to be sufficient for the scope of this project. The medium model consists of 24 decoder blocks, double the amount of the small version, as shown in the comparison above. This should arguably make an impact, as these transformer decoder blocks are the mechanism in charge of the text-generation process.

Using the module `aitextgen`, a pre-trained version of GPT-2 medium was used for the upcoming experiments. The original model is trained on a large dataset of millions of webpages called *WebText*, which arguably can suit the task of generating user movie reviews quite well. After fine-tuning the model on the new dataset, the model was saved and inference using the dedicated `generate` function could be performed.

5 Experiments and Results

This section describes how the experiments of movie review generation were conducted. First, the experimental setup is described, before the experimental results are presented for the following discussion to come.

5.1 Experimental setup

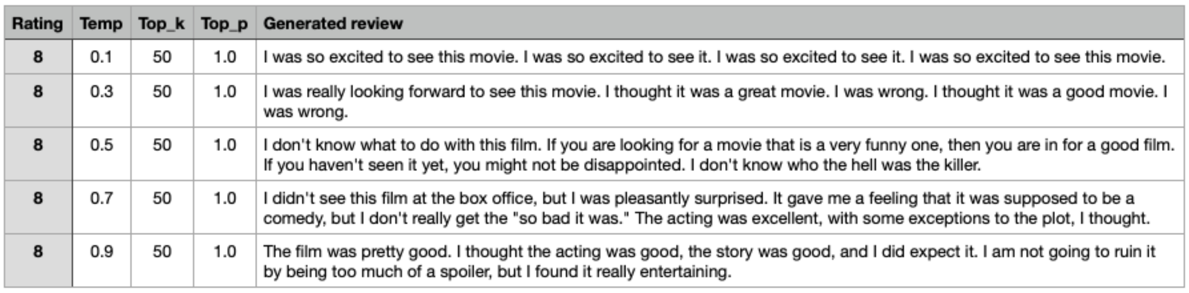
The experiments in this paper explore the impact of each hyperparameter, with a sequential run through the parameters `temperature`, `top_k` sampling, and `top_p` sampling.

The first of the mentioned parameters, *temperature*, determines how greedy the generative model is. For a text generation task, that is, the randomness or how safe the selection of the next words is. If the temperature is low, there is a low chance of sampling the lower probability words, and the model will output the safest, but rather boring, text with small variation. On the other side, if the temperature is high, the model can output, with rather high probability, other words than those with the highest probability. The generated text will be more diverse, but there is a higher possibility of grammar mistakes and the generation of nonsense. In a movie user review case, the formality is not necessarily the most important factor, and textual creativity can be appreciated, thus allowing the temperature hyperparameter of having a rather high value. The two additional parameters, being the top_k and top_p sampling hyperparameter, controls the behavior described in Section 2.

The default value of each parameter is set by `aitextgen` and the underlying `Hugging Face Transformer` modules, and equals 0.7 for temperature, $k = 50$ for Top-k sampling, and 1.0 for Top-p sampling.

5.2 Experimental results

5.2.1 Temperature



Rating	Temp	Top_k	Top_p	Generated review
8	0.1	50	1.0	I was so excited to see this movie. I was so excited to see it. I was so excited to see it. I was so excited to see this movie.
8	0.3	50	1.0	I was really looking forward to see this movie. I thought it was a great movie. I was wrong. I thought it was a good movie. I was wrong.
8	0.5	50	1.0	I don't know what to do with this film. If you are looking for a movie that is a very funny one, then you are in for a good film. If you haven't seen it yet, you might not be disappointed. I don't know who the hell was the killer.
8	0.7	50	1.0	I didn't see this film at the box office, but I was pleasantly surprised. It gave me a feeling that it was supposed to be a comedy, but I don't really get the "so bad it was." The acting was excellent, with some exceptions to the plot, I thought.
8	0.9	50	1.0	The film was pretty good. I thought the acting was good, the story was good, and I did expect it. I am not going to ruin it by being too much of a spoiler, but I found it really entertaining.

Table 1: The output of the `temperature` hyperparameter experiments

The first hyperparameter to be tested was the temperature. A selection of 5 values was taken into consideration, being 0.1, 0.3, 0.5, 0.7, and 0.9. These should be able to show the impact of the hy-

[12]`https://github.com/MartinJohannesNilsen/TDT12-Computational-Creativity`

perparameter on the model's ability to generate a good review. The other two hyperparameters, Top_k and Top_p, were set to their respective default values. Table 1 presents a manually curated selection of reviews generated by the model. These are truncated, as the length was set to be in the range of 100 to 300 words.

5.2.2 Top-K sampling

Rating	Top_k	Top_p	Temp	Generated review
6	10	1.0	0.7	This film is not a good movie, but it is good enough to be good. The acting is good, the cinematography is good, the score is mediocre, and the ending is good, the story is good, the cinematography is good, the score is good, it is entertaining and the cinematography is good. I can't fault this movie for the performances, but the storyline is good, the effects, the score is good
6	30	1.0	0.7	I am a big fan of the film, and I loved the film. I thought it was a good way to portray a very powerful character. But the story line was just about a man who is trying to get a job to fix a crime. I don't mind the film, but I think I was wrong.
6	50	1.0	0.7	This movie would have been better if they had been so much more than $5 million to get to the hotel. I was struck by how bad this movie is. I couldn't say it was the worst movie I've ever seen. I've seen better movies. I will have to mention that I did not like it either. I will never see it again. I gave it a 5.5 because it's a movie that's too much for a horror movie. I can't blame the actors for their performances. They did a very good job. They had enough time to do a great job. I wasn't sure who was the killer. It wasn't clear who was who the killer. Who knows?

Table 2: The output of top_k hyperparameter experiments

Next up is the hyperparameter for Top-k sampling. Presented in Table 2 are three rows of generated reviews. The first row has the lowest value of k, meaning that the model had the lowest amount of words to sample from. This should make the repetition of certain words almost inevitable. This is indeed the case, seeing the repetition of the word 'good' in every sentence, additionally multiple times in the same sentence. The generated review in the second row also includes some repetition, here in the form of the word "film", but the repetition is not as significant as in the row above. In the last generated review, the higher amount of possible next words to select from should make the model better suited for generating more varied text. This will be further discussed in Section 6.

5.2.3 Nucleus sampling

Rating	Top_p	Top_k	Temp	Generated review
4	-	20	0.7	I don't know what to do with this movie. I don't know what to do with the movie. [loop]
4	-	40	0.7	I don't know what to do with this movie. I don't know what to do with the movie. [loop]
4	-	50	0.6	I don't know what to do with this movie. I don't know what to do with the movie. [loop]
4	-	50	0.7	I don't know what to do with this movie. I don't know what to do with the movie. [loop]
4	-	50	0.9	I don't know what to do with this movie. I don't know what to do with the movie. [loop]
4	0.6	50	0.7	[...] I don't know what the hell was the deal with the plot. I was not sure what the hell was going on with the movie. [loop]
4	0.75	50	0.7	I am a fan of this movie, and I have a problem with the movie. The movie was a bit overrated [...]
4	0.9	50	0.7	I'm sure they are trying to make a good movie. The story is good, but the acting is terrible.

Table 3: The output of top_p hyperparameter experiments

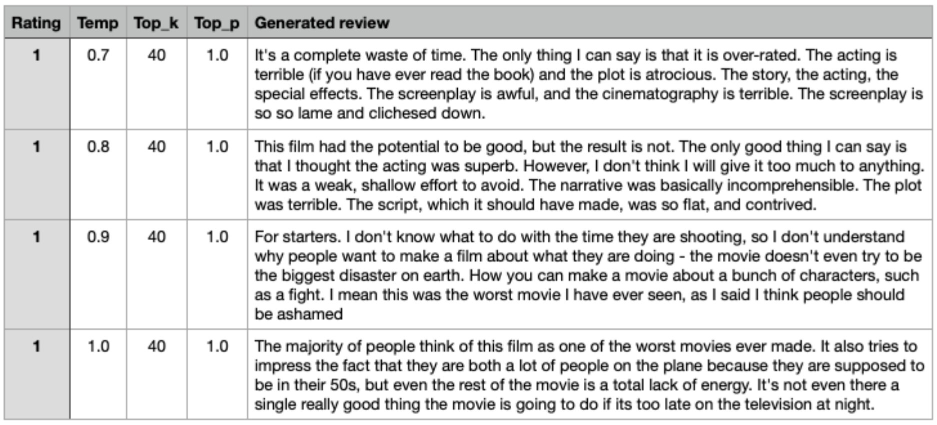
As the default value of Top-p is 1.0, as previously mentioned, nucleus sampling is used in its most extreme form in both of the two previous experiments. Therefore, it might be interesting to see the effect of turning off the nucleus sampling, in addition to showing the variation using different values for p. As one could observe by Table 3, the first 5 rows show the generated reviews when nucleus sampling was deactivated. The reviews seem to have been stuck in a repetitive loop, no matter the value of either Top-k or temperature. By including nucleus sampling, the generated text seems to take a step in the right direction, although the review with $p = 0.6$ also goes into a repetitive loop later on. These repetitive loops are not present in the last two rows, which might illustrate why the optimal value presented in Section 2 was 0.75. It should also be noted that the original paper mentions the use of $p = 0.95$, which seems to be a good choice based on the presented reviews above.

6 Evaluation and Discussion

The discussion part of this paper is divided into two parts. For the first part of the discussion, the impact of each hyperparameter mentioned in the last section will be described. Following this, the model's ability to generate a suitable review of the prompted rating will be discussed.

6.1 The impact of each hyperparameter

Starting with the temperature, Table 1 shows a significant change in behavior from temperature equal 0.3 to 0.5. The first of these seems to repeat the phrase ”I thought it was a great movie. I was wrong.” multiple times, which is seen as a safe option, rather than generating a better flowing text. The review where temperature equals 0.5, on the other hand, is arguably much better written. Distinguishing the two final rows in Table 1 is a harder task, which shows why the default value of temperature is set to be 0.7 in the `aitextgen` module. Based on the observations, a new run of tests was executed, resulting in the reviews presented in Table 4.

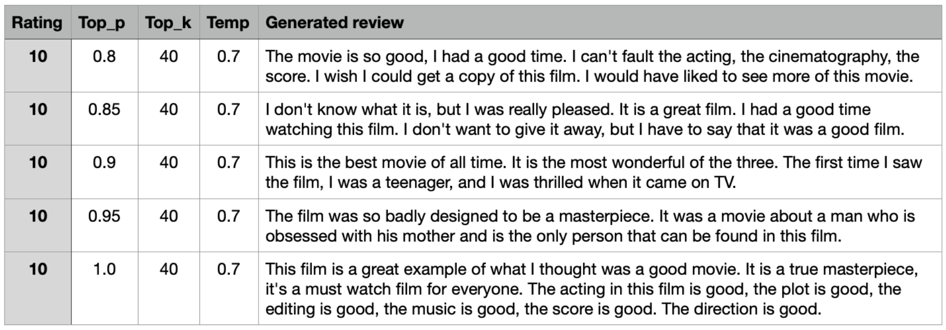


Rating	Temp	Top_k	Top_p	Generated review
1	0.7	40	1.0	It's a complete waste of time. The only thing I can say is that it is over-rated. The acting is terrible (if you have ever read the book) and the plot is atrocious. The story, the acting, the special effects. The screenplay is awful, and the cinematography is terrible. The screenplay is so so lame and clichesed down.
1	0.8	40	1.0	This film had the potential to be good, but the result is not. The only good thing I can say is that I thought the acting was superb. However, I don't think I will give it too much to anything. It was a weak, shallow effort to avoid. The narrative was basically incomprehensible. The plot was terrible. The script, which it should have made, was so flat, and contrived.
1	0.9	40	1.0	For starters. I don't know what to do with the time they are shooting, so I don't understand why people want to make a film about what they are doing - the movie doesn't even try to be the biggest disaster on earth. How you can make a movie about a bunch of characters, such as a fight. I mean this was the worst movie I have ever seen, as I said I think people should be ashamed
1	1.0	40	1.0	The majority of people think of this film as one of the worst movies ever made. It also tries to impress the fact that they are both a lot of people on the plane because they are supposed to be in their 50s, but even the rest of the movie is a total lack of energy. It's not even there a single really good thing the movie is going to do if its too late on the television at night.

Table 4: The output of further investigating the temperature hyperparameter

As one could observe from the table above, there is a fine line between writing an interesting review with a good flow, not only taking safe and boring choices all the time, and writing a review that does not make sense. For a temperature equal 1.0, the review could arguably suit a 1-star review, but it is not well put together. When compared to the second-row review, one could argue that the sentences in the second one make more sense, and therefore make up a better review overall.

Comparing the different reviews of Top-k sampling, the repetition of words does seem to decrease as the value increase, as previously mentioned. This makes sense, as the selection of words to sample from increases with the value of k. Looking at the table, one could argue that a value in-between 30 and 50 could be a good way to go. By reading the literature, popular choices seem to be either 40 or 50, which supports this statement, in addition to the default value by `aitextgen` being 50.



Rating	Top_p	Top_k	Temp	Generated review
10	0.8	40	0.7	The movie is so good, I had a good time. I can't fault the acting, the cinematography, the score. I wish I could get a copy of this film. I would have liked to see more of this movie.
10	0.85	40	0.7	I don't know what it is, but I was really pleased. It is a great film. I had a good time watching this film. I don't want to give it away, but I have to say that it was a good film.
10	0.9	40	0.7	This is the best movie of all time. It is the most wonderful of the three. The first time I saw the film, I was a teenager, and I was thrilled when it came on TV.
10	0.95	40	0.7	The film was so badly designed to be a masterpiece. It was a movie about a man who is obsessed with his mother and is the only person that can be found in this film.
10	1.0	40	0.7	This film is a great example of what I thought was a good movie. It is a true masterpiece, it's a must watch film for everyone. The acting in this film is good, the plot is good, the editing is good, the music is good, the score is good. The direction is good.

Table 5: The output of further investigating the Top-p hyperparameter

Based on the observations on the Top-p hyperparameter in the former section, one final run of generation was performed for being able to distinguish between the values. The hyperparameter was tested in the range of 0.8 to 1.0, with a step of 0.05. Based on the generated reviews presented in Table 5, one could argue that no choice is better than the other. The selection of one of these values would indeed generate good quality outputs and is therefore seen as valid values.

6.2 Does the review match the prompted rating?

The final question to raise is whether or not the generated reviews tend to suit the rating it is supposed to represent. Based on this, a new run of generation was performed using what was thought to be the best setup of hyperparameters, before two reviews were manually picked out. During the work on this project, another project was done in parallel, with the task of classifying movie reviews into movie ratings. This model, using RoBERTa, was therefore used for the sake of testing what movie rating it would output. This could then be compared with the prompted rating.

> *"This is a bad movie, it's not bad enough to make this movie the worst movie i have ever ever seen. Strike that, it was the worst I've ever seen. I am not saying this is a bad movie. It is an awful movie. And a major disappointment. The story is not very well told, the acting is horrible. The music is a typical bad joke. The directing is atrocious, horrible and amateurish. The ending is terrible. I wish I could give this movie a zero, but it was a disappointment. I don't know where it came from. It's not entertaining. It's not worth seeing."*

The review above can quite easily be classified as a bad review, showing quite clear negative sentiment. Feeding the review to the described RoBERTa model, the model outputs a prediction of 1.33. This does match the prompted 1-star rating and is a promising result.

> *"What a great movie! The scenes are not the only good thing about this movie. I thought the movie is awesome movie. I really was laughing at the beginning of the movie, the ending was excellent. I was not a fan of the story, but the ending was well acted. This is a good movie for anyone in a small age."*

The second review was created using the prompt of a 9-star rating. Taking a look at the output review, the sentiment seems to be very positive all the way through before the part of the sentence "I was not a fan of the story", which nudges the classification in the negative direction. Following this, the rest of the text has a positive sentiment again, and one could argue that the review does match a 9-star review. Feeding it to the classification model, the model classified it as a 9-star review with a score of 8.7, validating our thoughts.

7 Conclusion and Future Work

For concluding this report, one could raise the question of whether it is possible to make good movie reviews based on a prompted movie rating. Evaluating what is "good" is not always straightforward. In this paper, we have focused on the generated text's ability to be mistaken for human-like text. This is strictly based on text flow, with a subjective answer, and in need of manual, human, control. The lack of an automated evaluation metric like a BLEU score is crucial. BLEU is intended for machine translation and needs an annotated text to be seen as the correct answer, which is not possible to generate for this use case. As every movie review is different, there does simply exist no correct answer. One thing it can be evaluated on, nevertheless, is how well it suits the rating it is intended to fit. As we saw in the former section, the movie review classification algorithm developed in a parallel project did in fact correctly classify each of the two quoted reviews. Though, worth mentioning is the fact that these are manually picked out by a human subjectively meaning these are good written reviews. Not all generated reviews are perfect, in fact, some turn out to be garbage, and only the best picks were even bothered being checked.

Talking about the potential for improvement, one major nudge can be sent in the direction of the dataset. With the fact that the GPT-2 model is trained on over 8 million documents, our 70.000 reviews could advantageously be expanded even more. This also made a massive difference for the base GPT-2 model over the original one. The creation of a new, larger, and more balanced dataset, with all ten ratings being equally represented, could be a huge contribution to the field. If this were sought to be done, a focus on uniform text preprocessing and cleaning would also be beneficial. During the process of exploring the dataset, another problem stood out to us.

> *"This is definitely one of the best Kung fu movies in the history of Cinema. The screenplay is really well done (which is not often the case for this type of movies) and you can see that Chuck (in one of his first role) is a great actor. The final fight with the sherif deputy in the bullring is a masterpiece!"*

This review got a 1 out of 10 rating, which obviously is a mistake. This will impact the generation of reviews, as it can make it hard for the model to distinguish between good and bad sentiment, which is the most important part of a review suiting a low or high rating. Multiple of these reviews exist, where the rating does not match the review being written. If a better dataset were to be created, making sure these mistakes are not made would make an impact on the model's ability to generate good reviews based on a prompted rating.

References

Jay Alammar. The illustrated gpt-2 [blog post]. 2018a. URL `https://jalammar.github.io/illustrated-gpt2/`.

Jay Alammar. The illustrated transformer [blog post]. 2018b. URL `https://jalammar.github.io/illustrated-transformer/`.

David Bamman, Brendan O'Connor, and Noah A Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, 2013.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://aclanthology.org/E17-1059`.

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on cbr. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 33–46. Springer, 2004.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

Hanhoon Kang, Seong Joon Yoo, and Dongil Han. Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, 39(5):6000–6010, 2012.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2004. URL `https://aclanthology.org/S14-2004`.

Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.