ELEC-E5550 - Statistical Natural Language Processing
Movie Generator

Tudor Nicolae Mateiu Alicia Núñez Alcover

April 29, 2021

Abstract

Our proposal for the project of the SNLP course is that of a two-part movie generator. On one hand, we will be using a pre-trained GPT-2 architecture to fine-tune it using datasets containing movie plot descriptions in order to obtain newly generated movie descriptions depending on word input into the model. On the other hand, we will go one step further by using an additional architecture trained for text summarizing to obtain movie titles based on what our description generator architecture outputs.

Contents

1 Introduction

Natural Language Generation (NLG) is a swiftly developing field that can be defined as the task of automatically generating new text designed to appear indistinguishable to human-like written texts. It is an important key in the role of many natural language tasks such as long text generation, language translation, question answering, summarization, among others. Building these language models is a difficult task: generated texts must be coherent, grammatically correct, natural-looking and semantically meaningful.

In this project we focus on the study of two different methodologies and their effectiveness, we specifically focus on the methods for carrying out the NLG tasks of **long-text generation** and **text summarization**, as well as the efficacy of different evaluation methods and metrics. To achieve this task, we generate movie descriptions and a title is created based on the description generated.

Text generation remains an open and subjective task even nowadays, but revolves around the idea of having a machine generate completely new text given a few words for context or to begin the sentence. On the other hand, generally, text summarization can be split into two techniques: extractive and abstractive. Extractive summarization methods identifies important sentences verbatim from the text, without generating new text. Abstractive summarization interprets the text and generates a new summary text without repeating them verbatim.

Title: Terminator

A human soldier is sent from 2029 to 1984 to stop an almost indestructible cyborg killing machine, sent from the same year, which has been programmed to execute a young woman whose unborn son is the key to humanity's future salvation.

An extractive summarization model could summarize the text as "*Human soldier*" or "*Cyborg killing machine*", which does not catch any attention, whereas abstractive summarization models are more creative and could summarize the movie as "*The Executioner*". We are aiming at generating creative and original titles, and therefore, abstract summarization is undoubtedly the most suitable approach.

Traditional approaches tackled this task employing **recurrent neural networks (RNNs)** and **long short-term memory (LSTMs)** [1] which have been considered the state-of-the-art approaches. Many other ideas emerged to propel the efficacy of recurrent models by combining them into an **encoder-decoder** model [2], but these approaches do not succeed at retaining the context across longer sentences. Adding **attention mechanisms** [3] to these networks solved this problem, which identify relationships between words regardless of their distances, resulting in models performing better in longer input sequences.

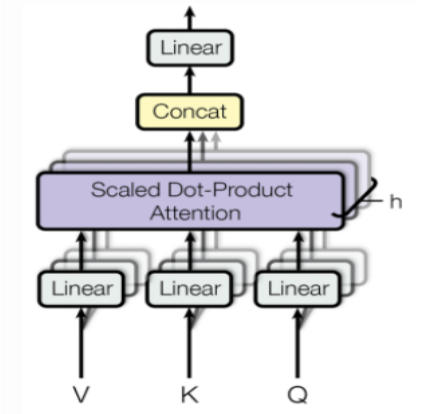


Figure 1: One such type of attention, a multi-head scaled dot-product attention mechanism [4].

However, the main burden of these networks has always been there: RNNs are inherently sequential, which means that their process cannot be parallelized, therefore constraining the length of the sequences as it would critically increase the training and inference time. The recent **transformer-based models** [4] caught our attention, as they have been replacing the traditional approaches [5] by relying only on the attention mechanisms. Therefore, transformer models allow efficient parallel training and models can be trained on large and generic corpora, which has led to incredible improvements on many language modeling tasks.

A transformer is an encoder-decoder architecture, where the encoder is composed of encoding layers that processes the input to generate encodings and the decoder is composed of a set of decoding layers that does the opposite, using the contextual information to finally generate an output sequence. To accomplish this, each layer makes use of an attention mechanism (self-attention), and, for each input, weighs the relevance of every other input, creating contextual information to finally produce the output. Then, the output of these attention mechanisms are fed to a feed-forward neural network. Self-attention can be defined as a layer that helps the encoder look at other words in the input sentence as it encodes a specific word for context purpose.

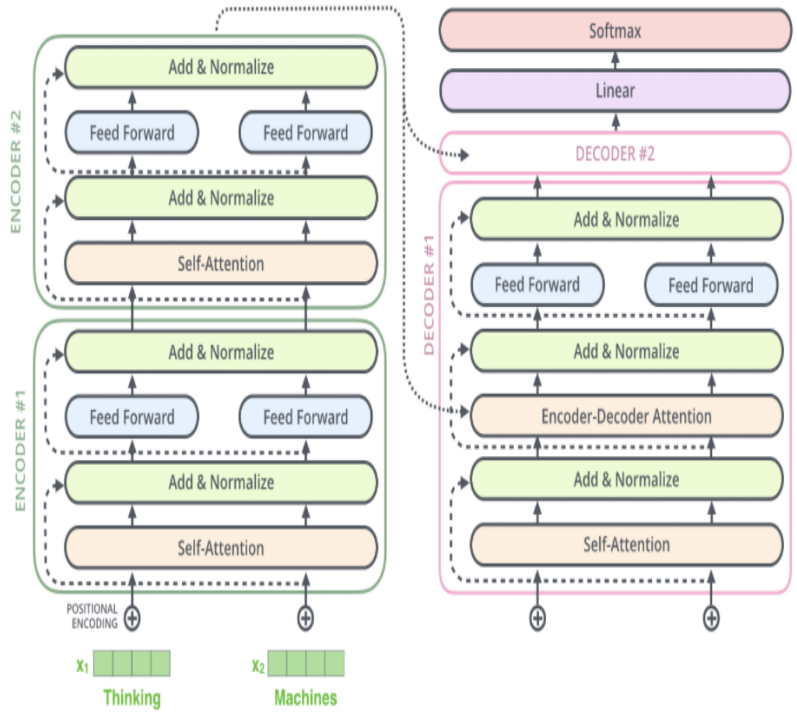


Figure 2: Example of encoder-decoder architecture with attention [6].

Pre-trained transformer-based models have arisen such as **BERT** (Bidirectional Encoder Representations from Transformers) [7] and **GPT** (Generative Pre-trained Transformer) [8], models that have been trained with a huge volume of data and adapted to be fine-tuned for specific tasks.

Specifically, we focus on the GPT-2 [9] model, which is a 'direct scale-up' of GPT. GPT-2 was trained on a massive dataset containing 8 million web pages, emphasizing on diversity and content quality. GPT-2 has achieved state-of-the-art on specific language modeling tasks (e.g. LAMBADA) and has proven to be able of generating coherent human-like written text. On the other hand, BERT was trained on a large dataset containing the entirety of Wikipedia and a book corpus and it achieved state-of-the-art performance on different NLP tasks such as Question Answering (SquAD v1.1 and v2.0), Natural Language Understanding (GLUE) and others.

Long-text generation results remain an uncharted territory. The main issue of text generation is the lack of control. Given the nature of GPT-2's architecture, we are able to generate consistent text and in concordance with the text style of the dataset. On the other hand, although BERT provides successful results in summarization while being efficient and faster than any traditional approach thus far, we will make use of BART, an improved sequence-to-sequence architecture that has as a basis the merger of GPT's auto-regressive features with BERT's transformers and obtains better results at abstractive summarization [10].

Given that long-text generation and text summarization can be very open-ended and subjective, for example the same text input can generate practically an infinite amount of different outputs or summarized in a multitude of ways. Because of this, finding a suitable metric that captures all the quality aspects of a generated text is a challenging undertaking, where human evaluation is the best method to guarantee a high-quality generated text [11]. However, using human evaluation can be unfeasible in just about all occasions, therefore it is normal to seek out other quantifiable automatic metrics. All of these NLG metrics are usually grouped [12] into three categories:

- **Human Evaluation.** The most instinctive and logical way of evaluating NLG is by using human judgement to rate or compare texts generated by NLG's or to carry out a Turing test to distinguish these generated texts from human generated ones. While this evaluation gives the finest judgement on a model's performance, it can also become quite expensive and time-consuming, additionally inconsistencies can arise between experiments and human evaluators, which may prevent the reproduction of result comparisons and assessments [13]. Although there is no consensus in how human evaluation should be performed, there are many insightful current practices that are taken into account in this project [14].

- **Untrained Automatic Metrics.** Many automated metrics have been made to allow fast and cheap evaluation. These are the most commonly used metrics, which compare machine generated texts to human ones generated on the same input data. These metrics calculate a score which indicates the similarity between the generated text and input text. These metrics are in turn grouped [12] into five categories, but out of which we will use the Perplexity and the ROUGE metrics from the so-called *n-gram overlap metrics*.

  Perplexity [15] is the second best choice for evaluating the output quality of long-text generation models, as so far the first choice is human evaluation, and the others, e.g. BLEU [16], compare the output text to the input text, which wouldn't be realistic for use on the task at hand.

  ROUGE [12] is a widely used metric, specifically designed for the evaluation of NLG automatic text summarization, and the results obtained in recent publications [17] [18] [19] all point to it leading the way in evaluation of text summaries. ROUGE measures the quality of a text by computing the frequency of overlapping n-grams. The ROUGE package offers many variants of this metric depending on the n-gram length, but ROUGE scores are heavily based on lexical similarities and it does not take into account the fluency of the generated summaries and thus, this metric can not be suitable for our abstractive summarization task. Although many ROUGE variants exist for synonyms and paraphrasing, such as ROUGE-WE [20], ROUGE is still considered the default metric for automatic text summarization, leaving us with no actual metric that is satisfactory for this task, and, therefore, our last resort is performing human evaluation as well.

- **Machine-Learned Metrics.** These metrics work in a similar fashion as human-centric evaluation metrics, in that a judge evaluates machine generated text, but instead of a human it's a machine-learned model. They will not be considered in this project given the sharp increase in difficulty to implement them.

Our approach is to delve deeper into two transformer models (GPT-2 and BART) to witness their efficiency in our project, by working in two different subtasks: long-text generation and text summarization. Our aim is to first preprocess the data collected and then procure pre-trained GPT-2 and BART

models and fine-tune them with movie descriptions and movie titles respectively. Additionally, for long-text generation we aim to analyze the correlation between the perplexity scores and human evaluation. For text summarization, because of the problem of summary brevity, we will carry out evaluations via human judgement.

2 Methods

In this section we describe the different methods that we carried out during the course of the project to achieve the end goal we desired. These range from the study and implementation of the models used, to data preprocessing and preparation for the experiments and finally to their respective evaluation and interpretation.

2.1 Data preprocessing

To carry out this task, we have gathered different datasets that contain a diversity of movies and TV show descriptions. The TMDB Movies, Netflix Movies, Anime Data and Movies Metadata datasets have been used, which contain movie information such as the title, description, genres, release year and cast of a total of 58570 movies and shows.

In order to create data suitable for training, each dataset has been filtered and cleaned to obtain a readable format. The following filters were applied via code:

- **Null value filter**: logically, movies that contained no description or title were altogether removed.
- **Duplicate value filter**: the uniqueness of movies in the data has been taken into account by removing duplicated movies found in all datasets.
- **Minimum length filter**: because some movie descriptions contained a handful of words and because of problems that appeared after having trained the description generator model, all movies with description lengths (in words) below 40 have been removed.
- **Non-Latin character filter**: movies that have non-Latin characters, such as Chinese, in either their description or the title have been removed in order to not cause unexpected errors during training or consequently obtain unwanted results such as combinations of non-Latin and Latin characters as outputs.

The dataset Anime Data contained many rich and creative descriptions of Japanese animation, but the titles were in Romaji (Latin script to write Japanese). This might have created a potential problem of titles fully or partially generated in Romaji, leading us to create two different datasets for training the movie description model and title generation model, where the title generation model simply has this dataset removed.

Finally, after having selected, filtered and cleaned the datasets by use of the filters, two new datasets were created for each model: one containing a final count of 31084 movies for training the description generator, and one containing a final count of 29782 movies for training the title generator (with the anime dataset removed).

2.2 GPT-2: Movie description generator

As we know, the original transformer was composed of encoder and decoder stacks, whereas GPT-2's architecture is composed of transformer decoder blocks. One of the differences between GPT-2 and BERT is that GPT-2 is auto-regressive: it outputs one token at a time and that token is added to the input sequence, creating a new sequence that is fed to the input of the model in the next step.

GPT-2 uses self-attention in its decoder blocks, where each block is composed of multiple layers, containing a multi-headed masked self-attention operation and then, a fully connected layer. Masked self-attention, unlike self-attention, only takes into account previous tokens.

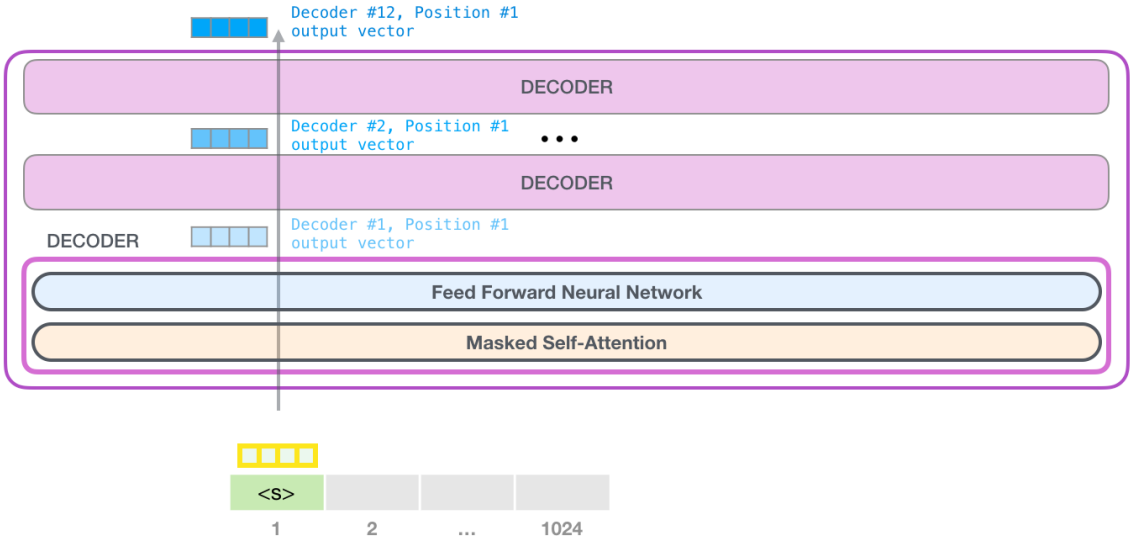


Figure 3: An example of a decoder block in the GPT-2 network [6]

Basically, the architecture receives one input, which is processed through all the layers and a vector is created along the model.

Moreover, in order to generate new movie descriptions, there are many decoding strategies to produce fluent and coherent text. We will explain the strategies covered in this project:

GPT-2 was trained four times with different number of parameters: 117M (small), 345M (medium), 762M (large) and 1.5B (xl) parameters. In the paper of GPT-2 [21], it has been shown that by increasing the number of parameter in the model, perplexity decreases. In our work, given the limitations such as the resources and time, we have chosen the medium model.

Basic sampling is a non deterministic decoding that outputs a randomly picked next work based on its conditional probability distribution via: $w_t \ P(w|w_{1:t-1})$ at timestep t. This method usually does not output very coherent text. A trick that is usually used is the parameter **temperature** which increases the likelihood of high probability words and decreases the likelihood of low probability words, this is made by lowering the "temperature" of the softmax function.

Top-K sampling consists in selecting the top-k most likely next words and the probability mass is redistributed to these k words. This method allows us to generate more coherent text, but there are concerns with this technique as it can eliminate the possibility to sample words that can be reasonable choices and can produce gibberish. This has led to create **Top-p sampling**, which instead of choosing the most likely K words, it chooses from the smallest set of words whose probability is larger than p and then the probability mass is redistributed to the words in this set.

As a general rule in open-ended language generations, it seems that top-p and top-K can produce fluent and coherent text, more than other traditional methods such as greedy or beam search, although it has been shown that can suffer from generating repetitive word sequences [22]. Therefore, we must find out which parameters are the most suitable for our task in the experiments.

2.3 BART: Title generator

BART [23] is a denoising autoencoder, presented by a team from Facebook AI, that has been trained by corrupting text documents and then making the model learn to reconstruct the original text. It uses a standard sequence-to-sequence Transformer architecture, similar to that of BERT, by using a BERT bidirectional encoder and additionally employing a GPT left-to-right auto-regressive decoder.

BERT is pre-trained to predict masked tokens, and uses the full sequence to get sufficient information to make a good prediction. On the other hand, GPT-2 is pre-trained to predict the next word and is thus more successful for generation tasks. Because of this merge of BERT and GPT architectures, the BART model gets the benefits of the two widely different architectures, and thus it can fine-tuned for the task of summarization, more specifically, for our goal of obtaining movie titles via abstractive text summarization of movie descriptions.

We make use of the Simple Transformers library for Python, which is a community developed and lead library based on the Transformers library by Hugging Face, to carry out the download of Facebook AI's pre-trained BART model and then fine-tune it for our needs. It is noteworthy that there are multiple BART models available that range in different number of parameters, we concretely use the base model.

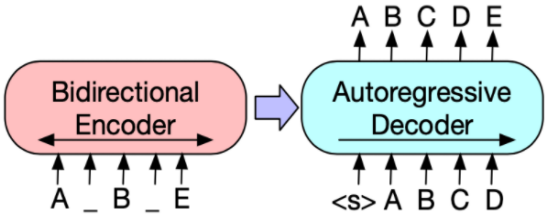


Figure 4: On the left, a corrupted document is encoded with one of the bidirectional encoders, and then, on the right, the likelihood of the original document is computed with one of the auto-regressive decoders. [23].

2.4 Evaluation metrics

The first step in the evaluation stage was to calculate the perplexity of the generated descriptions. the second step entailed human evaluation in order to analyze the correlation between the perplexity values and the human evaluation results. Moreover, to evaluate the title generations, human evaluation was carried out as well, given that the attempt to use automated metrics such as ROUGE was unsuccessful since the task generates abstract small titles and thus, it cannot be measured automatically.

2.4.1 Perplexity

GPT-2 architecture uses cross-entropy as the loss function between the training data and the predictions. Cross-entropy measures the performance of a model by computing the difference between two probability distributions P and Q for a given random variable. For discrete probability distribution we have the following equation:

$$H(P,Q) = -\sum_{x} P(X=x) log Q(X=x)$$

where P is equal to the word distribution of the dataset and Q is equal to the predicted output for the next word, given the context. In the case of a continuous distribution, a similar calculation is made by assuming that P and Q are absolutely continuous and a integration is made, instead of a sum.

Then, perplexity, which measures how well a model or probability predicts a sample and it can be used to compare probability models. A low perplexity value indicates that the probability distribution is

good at prediction the sample. Perplexity can be computed as follows:

$$PPL = 2^{H(P,Q)} = 2^{-\sum_x P(X=x) log Q(X=x)}$$

Consequently, since the model minimizes the cross-entropy loss, it is as well minimizing the perplexity value.

2.4.2 Human evaluation

The human evaluation metrics in this project have been based on the guidelines given in the article [14]. To get an insightful evaluation we focus on the quality of summaries through direct human judgment using different criteria. These criteria should be selected in concordance with the type of text that we are generating, for instance, in a summarization task we seek correctness and readability.

The generated movie descriptions were evaluated on three criteria: coherence, creativity and credibility.

- **Coherence**: this metric will measure how consistent the description is. This means that for a high value of coherence, the context between sentences and words holds strong.
- **Creativity**: this metric will measure how original the description is. It involves things such as inventive names for people, things and places, good use of acronyms and quotation marks, and plot ideas either fully or partially original
- **Credibility**: this metric will simply measure how much is the description perceived as artificially created instead of having been written by a human.

The generated titles were evaluated on two criteria:

- **Relevance**: this metric will measure how relevant the words generated for the title are in relation to the movie's description. A low value would mean that the title is irrelevant when considering the description's context, i.e. using words that have nothing to do with the description.
- **Creativity**: this metric is closely related to the same used for descriptions, but by using this metric for titles we more specifically try to measure the amount of abstractness of the titles generated. A high value of creativity would mean that the title describes the movie with a fair ratio of original words, instead of extracting most, or even all, words directly from the movie description, which would entail a lower value of creativity. Additionally, the creativity can be high if the title's individual words are extracted from the description but they are written out in an original manner.

There are a wide range of rating methods that are used in human evaluation. The most popular method is categorical choice such as yes/no or preference rankings whereas Likert scales are in overall the most popular method. Likert scales are a type of response scale in which responders specify the level of agreement with an statement given. There are many different types such as 3-point, 5-point and 6-point. Moreover, instead of measuring statements of agreement we can measure different variations such as frequency, importance or quality.

The most suitable ranking for our task is the 5-point Likert scale that measures the quality of the generated movie descriptions that goes from best to worst, specifically we have used the scale: Excellent, Good, Acceptable, Poor and Very Poor.

2.4.3 Perplexity and Human evaluation

As we know, GPT-2 model minimizes the perplexity but it is well-know that perplexity does not correlate well, generally, with human evaluation in tasks of text generation such as generating TV Scripts [24]. In our project, we aim to analyze if there is any correlation between the perplexity values in our generations and the human evaluation performed.

3 Experiments

This section describes in detail the steps to obtain the movie descriptions and titles, how the models were fine-tuned and how to compute the different evaluations for these movies.

3.1 Environment Setup

The environment used to carry out all of the coding and experimentation is Google Colaboratory, a Jupyter notepad environment that makes available Python 3 runtimes with powerful Nvidia K80 and T4 GPU's with 12GB and 16GB of VRAM respectively.

This provides a powerful GPU environment for carrying out the training of the models as well as their evaluation and output generation. But it must be noted, that even so, the training times for both models have been excruciatingly high, more specifically for the description generator.

Unfortunately, Colab has a few limitations since it is free to use, a Colab session has limited use of running notebooks up to 12 hours and notebook instances can be shut down if they are running for many hours. Moreover, resources are available on the fly and vary over time to accommodate fluctuations of use in demand, meaning that your training could be interrupted at any time.

These limitations have been a set back in our experiments, as the training of a GPT-2 medium with approximately 31084 movies to train with, supposed ≈ 8 hours of training time and exhausting constant intervention in case the Colab environment disconnected, because of inactivity, CAPTCHA pop-ups or any other unexpected problems like the environment disconnecting without reason.

3.2 Data Setup

For the description generator, the data has been first loaded from the `.csv` file that contains the clean and merged datasets. Afterwards, the data is split in 85% for training and 15% for validation, and both sets are loaded into text files. Finally, the `transformers.TextDataset` function is called for the two text files in order to tokenize them (the tokenizer used is mentioned in the following section) in blocks of a maximum of `block_size = 128`. The data is then simply passed to the model for training.

For the title generator, there was no need to save the data in text files or tokenize it. The data was simply passed to the model for training.

3.3 Architectures Setup

In order to train the two architectures used for movie description generation and movie title abstractive summary we have used for each of them the following setup:

- **GPT-2 Description Generator:** for generating the descriptions with GPT-2 we have employed the Transformer library prepared by Hugging Face. Specifically, we have used the class `transformers.GPT2LMHeadModel` for initializing the model and then we call the model's class function `from_pretrained('gpt2-medium')` in order to load the medium sized pre-trained GPT-2 model offered by Open AI.

  Afterwards, before fine-tuning the model, we must first set up a `transformers.Trainer` which will receive as parameters the previously loaded GPT-2 model, a `transformers.TrainingArguments` class, a data collator function, the training dataset and an optional evaluation dataset.

  The data collator function is provided by the `transformers.DataCollatorForLanguageModeling` class, which first needs to be initialized with a tokenizer. For the tokenizer itself we used one that uses GPT-2's tokenization methods: `transformers.GPT2Tokenizer` and loaded a pre-trained one by calling its class function `from_pretrained('gpt2')`.

Finally, the `transformers.TrainingArguments` class that needs to be passed to the trainer we have initialized with the following parameters:

- `output_dir = 'gpt-model'`, the output directory for the fine-tuned model.
- `num_train_epochs = 8`, the number of training epochs.
- `per_device_train_batch_size = 8`, the batch size for training. It has been set this low because any value higher than this caused the Colab environment to crash due to insufficient memory needed to run the training.
- `per_device_eval_batch_size = 8`, the batch size for training. Set low for the same reason.
- `eval_steps = 500`, number of update steps between evaluations.
- `save_steps = 10000`, after this amount of steps the trainer will create a new checkpoint of the current state of training for the model. This was done in order to have checkpoints approximately every 2 hours, in order to resume training from there if the environment disconnected.
- `warmup_steps = 400`, number of warmup steps for learning rate scheduler.
- `evaluation_strategy = 'steps'`, this will carry out validation evaluation every amount of specified steps.

The model has been fine-tuned by simply calling `trainer.train()` with the created trainer above. Finally, after having fine-tuned and saved the model to Google Drive, in order to load it back we simply call again the same `transformers.GPT2LMHeadModel` class, but now by passing to the class function `from_pretrained(model_path)`, the path to the model after downloading and unzipping it from Drive.

- **BART Title Generator:** for summarizing the generated descriptions with BART we have employed Simple Transformers. For initializing the model with a pre-trained one, provided by the Facebook AI team, we specifically used the class `simpletransformers.seq2seq.Seq2SeqModel` by passing the following parameters to it in order to load the base BART model:
  - `encoder_decoder_type = 'bart'`
  - `encoder_decoder_name = 'facebook/bart-base'`
  - `args = model_args`

  Where `model_args` is a Python dictionary that contains the following parameters:

  - `'reprocess_input_data': False`, set to not waste memory or time on this step. Specifically set to False due to the memory instability of this library (as explained in the batch size parameter below).
  - `'overwrite_output_dir': True`, set to overwrite the output directory if there was one already created.
  - `'save_model_every_epoch': False`, set to not save checkpoints after every epoch.
  - `'save_eval_checkpoints': False`, set to not save checkpoints after evaluations.
  - `'save_steps': -1`, this parameter has been set to -1 in order to block the model from annoyingly saving checkpoints every 2000 steps. In total for training, the model runs 20000 steps, so it would save around 20 different checkpoints each weighing around 2GB. This was a real annoyance, and we believe it should have been automatically set to not save anything, as it would clog the disk space of the Colab environment and there was no need for checkpoints as the model needs around 40 minutes per epoch, not enough for the environment to disconnect due to any reason.

- `'max_seq_length': 100`, the maximum length of the summarized sequence.
- `'train_batch_size': 1`, batch size for training. The reason the value is so low is the same as for the description generator model, in order for the environment to not disconnect due to insufficient memory. A very important thing to note, we believe there may be problems of inefficiency or memory leaks with the library: it makes no sense for the GPT-2 medium model, which is around double the size of BART, to be able to run steps with a batch size of 8, and for this BART model to barely get by with a batch size of 1.
- `'num_train_epochs': 3`, number of training epochs. This amount of epochs was enough for the model to become abstractive in its summarization.
- `'cache_dir': 'cache/'`, cache directory for data during training. It's description in the documentation made us believe it was the same as if we were to use TensorFlow 2's cache functionality, used in order to save a lot of time during training, but it was not so and this parameter had no impact whatsover on the training time.

The model has been fine-tuned by simply calling `model.train_model(train_slice)` with the model above. Here, `train_slice` is a slice of size 20000 of the training dataset (after shuffling). This slice was necessary in order to be able to run the code for training, as numbers higher than this would simply crash the environment and disconnect it, we believe this is because of the same reason as above, the library is inefficient and has memory leaks.

Finally, after having fine-tuned and saved the model to Google Drive, in order to load it back we simply call again the same `simpletransformers.seq2seq.Seq2SeqModel` class, but now by only passing as parameters `encoder_decoder_type = 'bart'` and `encoder_decoder_name = 'outputs'`, where `'outputs'` is the name of the folder containing the model after downloading and unzipping it from Drive.

3.4 Generation Setup

In order to generate different movie descriptions, we have used the function `generate` from the GPT-2 model `GPT2LMHeadModel`. The function receives an input that needs to be tokenized and encoded beforehand by using the object `GPT2Tokenizer` and its function `encode`.

Given that we seek a broad variety of texts, we use different decoding strategies. We generate descriptions by looping through a combination of top-k sampling, top-p sampling and temperature parameters with different values. The range of values for top-k sampling was $[20, 50, 100, 200, 400]$, the range of values for top-p sampling was $[0.85, 0.9, 0.95, 1]$ and for temperature $[0.7, 0.8, 0.9, 1]$.

Moreover, as input for the generation function we have used a set of different inputs, with the purpose of getting creative and enriching descriptions:

- *After a zombie outbreak in Las Vegas*
- *It is the year 2077*
- *The human colony on Andromeda*
- *In the deep titanium mines on Alpha Centauri*
- *After a series of harrowing murders*
- *When five college kids arrive at a remote forest cabin*
- *During World War 2*
- *Humanity has mastered interplanetary spaceflight and begun to explore the galaxy*

Finally, this experiment generated 641 different movie descriptions with a wide range of perplexity values.

The next step was to generate different titles for our generated descriptions, which in this case we have used the function `predict` from our trained model BART. In this function is not necessary to tokenize and encode our descriptions beforehand.

3.5 Evaluation Setup

In order to evaluate our generations of movie descriptions and titles, we first evaluated the perplexity of each generated movie description and afterward, we created a public survey using Google Forms. In the survey, we chose 9 generated descriptions, where 2 descriptions were from low perplexity values, 2 descriptions from similar perplexity values to the test set, 2 descriptions that were cherry picked from us and 3 descriptions from high perplexity values. Each description contained the title generated as well. The reason behind cherry picking two descriptions without regard of the perplexity value was to evaluate if they could be considered good descriptions and could be even considered almost real descriptions. Moreover, the perplexity value obtained in the test set was 11.81 and thus, the similar perplexity set contained descriptions that were closer to this value.

In each movie, which contained the title and description, the participant was asked to rate the coherence, creativity, and credibility of the description from Excellent to Very poor. Moreover, it was also asked to rate the title according to the relevance and creativity, from Excellent to Very poor.

As a final question before submitting the form, the respondent was asked if they had any knowledge about language models, where the participant could answer 'Yes', 'No' or 'A bit'.

The survey was completely anonymous but there was no possibility to retake the survey again since in order to take the survey it was necessary to sign in to Google.

4 Results

This section shows the results obtained from the project. The results from the survey and perplexity values regarding movie descriptions can be shown in the section 4.2 whereas the results regarding the titles can be shown in section 4.3. The survey was in total answered by 21 different respondents.

4.1 Evaluation of movie descriptions

Figure 5 shows a density distribution of perplexity values over the 641 generations of movie descriptions, which varies in the range between 3.52 and 28.41. This plot uses a kernel density to show the probability distribution of the data. Moreover, it can be shown that the concentration of perplexity values lies somewhere between the values 5-10.

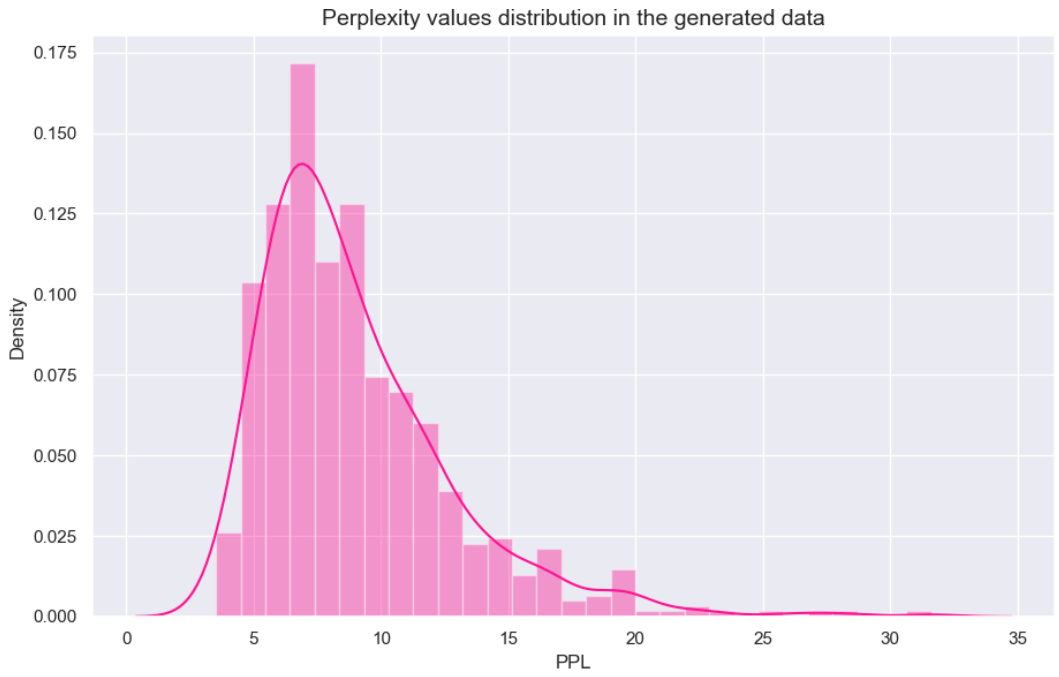


Figure 5: Perplexity values in the generated descriptions.

It can be seen that perplexity values widely vary, which it is expected as the generations were produced using different values of parameters and the descriptions significantly differ in length and content.

In the following two generated movies, the movie *2077* corresponds to a description with low perplexity whereas the movie *War of the Worlds* corresponds to a description with high perplexity. As it can be seen, the movie *War of the Worlds* is less predictable and the words used are more elaborated, which explains the high value of perplexity.

Title: 2077

It is the year 2077, and the human race has been driven to near-extinction by an extraterrestrial threat that is unknown to science. The only hope for mankind is the creation of a new type of weapon: a robot capable of destroying the threat.

Title: War of the Worlds

In the deep titanium mines on Alpha Centauri, massive robots wage war with lone astronauts, deadly radiation storms and a giant sentient drill. Overwhelmed by fighting machines and the threat of a homicidal robot uprising, Astronaut Jim Barclay discovers that his commanding officer is lying.

The results from the Google Forms survey regarding the generated descriptions are shown in Figure 6, which shows a histogram displaying the averaged values of Coherence, Creativity and Credibility for the low perplexity, similar perplexity, cherry-picked and high perplexity sets. In order to create a quantitative measure, the Likert scale used in the survey has been converted into a range of 1-5 values.

- **Excellent**: 5
- **Good**: 4
- **Acceptable**: 3
- **Poor**: 2
- **Very poor**: 1

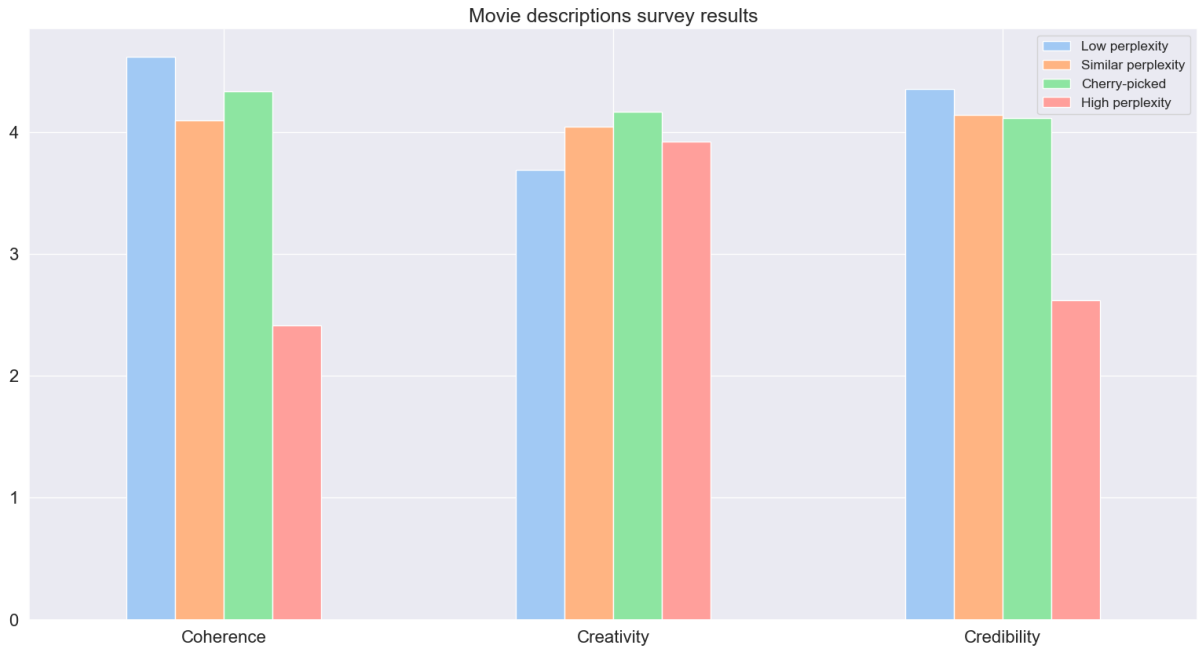


Figure 6: Movie description generations survey results.

The exact values of results from Figure 6 can be shown in the Table 4.1

Table 1: Movie description generations survey results.

Set	**Coherence**	**Creativity**	**Credibility**
Low PPL	4.62	3.69	4.36
Similar PPL	4.10	4.05	4.14
High PPL	2.41	3.92	2.62
Cherry-picked	4.33	4.17	4.12

As can be seen in Figure 6, we have arrived at the conclusion that the **Low** and **Similar** perplexity values are inversely dependent to the **Coherence** and **Credibility** metrics. With these values of perplexity, people tended to mainly vote that the descriptions were highly coherent and credible, the latter

being in actuality a logical consequence of the former: descriptions that are coherent don't stand out as fake.

This inverse dependency holds for **High** perplexity values as well, when perplexity increased, the **Coherence** and **Credibility** metrics decreased. This happens several times when the model, after having written some sentences that are coherent, suddenly starts writing about things that are not in the same context. The problem could be attributed to the minimum and maximum lengths set during training, and the model probably tries to reach these lengths by filling in the remaining sentences with ones with different context.

We have also noticed that with the **Low** and **Similar** perplexity values, the **Creativity** metric was usually lower, which would be the result of the model not taking chances at generating incoherent text or more elaborate words, or having reached the necessary length needed for the description. This is the opposite for **High** perplexity values, a higher value of perplexity correlates to higher **Creativity**.

The following movie is voted as the most creative by people who did the form:

Title: The Matrix

In the deep titanium mines on Alpha Centauri, scientists discover a fantastic alien invention: a mind matrix that can interface with any sentient device... including our bodies! But when the alien takes over the minds of local militia leaders, the military must investigate... and they find themselves battling a matrix that's out for blood!

And the following one is the most voted both for coherence and credibility:

Title: Dead Zone

After a zombie outbreak in Las Vegas, a group of survivors must cross the desert in an attempt to stay alive. Their only other contact is a mysterious radio tower that broadcasts a constant stream of cryptic messages.

4.2 Evaluation of movie titles

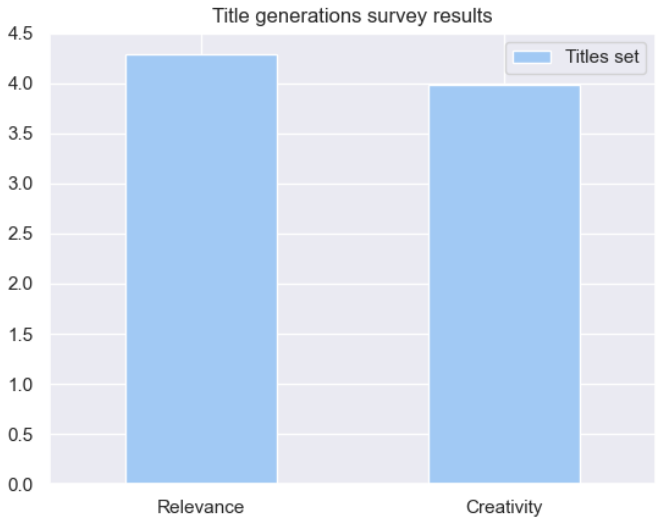


Figure 7: Title generations survey results.

The results from the Google Forms survey regarding the generated titles are shown in Figure 7.

The results from the form, and by analysing many other generated titles not present in the form, are exactly what we set out to accomplish regarding the title generator: an abstractive summary model. More concretely, the results in Figure 7 are as follows:

- The **Relevance** metric has a score of approximately 4.3 out of 5, this means that regardless if the summary is abstract or not, at least the summarized generated title has correlation with the movie's description.
- The **Creativity** metric has a score of approximately 4 out of 5, this means that the titles are all creatively obtained instead of directly taken words from the movie's description.

In addition to the results from the form, it must be noted that the title generator does have some problems. For instance, there are cases in which the title's words are taken straight from the description, but even so most times there are some additional changes, two examples of words taken literally from the descriptions (with and without additional changes):

Title: The Halsey Murders

After a series of harrowing murders in which a gang leader is savagely murdered and a local beauty is abducted, Inspector Jim Halsey finds evidence that points to a Mafia boss who has infiltrated his department - but he risks his own life by falling under the Mafia's tyrannical influence.

Title: Zentraedi

Humanity has mastered interplanetary spaceflight and begun to explore the galaxy. However, humanity's technological prowess has also brought new threats into the solar system, including a dangerous race of giant bugs. To combat these dangerous creatures, the government constructs the giant 'Zentraedi' robots.

This can be attributed to the descriptions containing names, dates, acronyms, words in quotation marks, etc, that might have more weight while generating the title than an abstractive title would. A clear example of this happens with the starter phrase *It is the year 2077*, where the generated title is mostly *2077*. In these instances we have further studied the phenomena and reached to the conclusion that these extractive generations have no connection to the perplexity of the movie's description, and additionally, most times when the description doesn't contain one of the previously mentioned tokens with high weight it mostly generates original abstractive titles.

Some examples of low (green) and high (red) perplexity descriptions with the same *2077* title:

Title: 2077

It is the year 2077, and the human race has been driven to near-extinction by an extraterrestrial threat that is unknown to science. The only hope for mankind is the creation of a new type of weapon: a robot capable of destroying the threat.

Title: 2077

It is the year 2077. Mankind is living in a primitive and inhospitable world. But the human kind has managed to conquer this environmental disaster and built cities. Humanity is on a rampage and the peace is slowly returning. A couple of soldiers on a space bus find a dying man in the desert.

Finally, it must be noted that most of the titles generated are quite distinct, creative and abstract:

Title: Robotropolis

In the deep titanium mines on Alpha Centauri, the crew discovers a mysterious robot. They hatch it, only to discover it's a deadly weapon designed to kill humans. As its killing spree escalates, they discover it's part of a massive plot to subjugate the human race... and all of space and time.

Title: Dead Zone

After a zombie outbreak in Las Vegas, a group of survivors must cross the desert in an attempt to stay alive. Their only other contact is a mysterious radio tower that broadcasts a constant stream of cryptic messages.

Title: The Dark Star

The human colony on Andromeda is decimated by the ferocious, flesh-eating creatures that roam the galaxy seeking human flesh. To save the human race, the crew of the colony ship "Phoenix" must fight the creatures and their deadly weapon in this thrilling sci-fi action-adventure.

Title: Venturi: The Last of the Human Zeta Warriors

The human colony on Andromeda is threatened by a new enemy, the horrific Venturi. Led by a sadistic villain, Venturi has enslaved a number of intelligent life forms aboard their ships in order to use them for their own sinister ends. The only hope for the survival of the human race is to liberate the last of the Venturi captives.

5 Conclusions and Discussion

In this project, we accomplished the task of developing a movie generator that uses two pre-trained architectures, and then fine-tune the models with the purpose of generating movie descriptions and obtaining an abstractive summary from the generated description.

We have used a GPT-2 and a BART architecture, as these transformer-based architectures have started to overcome the previously considered state-of-the-art RNN and LSTM approaches [5]. Given the nature of GPT-2's architecture, we are able to generate consistent text in concordance with the text style of the dataset, and BART will allow us to obtain better results at abstractive summarization than architectures such as BERT [10].

Given that text generation and text summarization are very open-ended and subjective tasks, finding suitable metrics that capture all the quality aspects of generated and summarized text for this particular task was a challenging undertaking.

For the task of generating text for the movie descriptions, we chose to use the perplexity metric for evaluating the output quality, as other metrics like BLEU wouldn't be realistic for use on the task at hand. Additionally to this, we decided to use human evaluation with the purpose of analyzing if there is any correlation between the perplexity values in our generations and the human evaluation performed, which in turn was done with the aim of measuring the coherence, creativity and credibility of the descriptions.

Although there is no clear relationship between perplexity and human evaluation, we can still conclude that coherence and credibility are tightly related to perplexity, since the model is able to write predictable

text and thus, more human-like. Moreover, as perplexity increases, coherence and credibility decrease but creativity increases as well. This behavior was expected from the start, as there is a compromise between coherence/credibility and creativity, we cannot expect that a language model is able to write elaborate generated texts without losing coherence and therefore, credibility.

In our task of abstract summarization using BART, we endured a few drawbacks but the most notable was the lack of automated metrics for abstract summarization. In our initial research, the idea was to find a suitable metric similar to ROUGE that would be able to measure this summarization, without luck. As a last resort, we used human evaluation in the same fashion as in the procedure followed in the evaluation of movie descriptions, which gave us a significant insight regarding the quality of the titles generated by our model.

In the future, we hope for advancement in the creation of automated metrics that are suitable for tasks such as text generation, where we expect unsupervised creative and coherent texts, or in tasks related to abstract summarization such as in the generation of movie titles, where the creativity is an important focus to catch the attention of the viewers. Right now, the most suitable evaluation method is human evaluation, which gives us the finest judgment on a model's performance, but it is also quite expensive and time-consuming, in addition to the subjectiveness, which leads to many different evaluations in the same task.

Possible future work, given that resources and time were the main limitations in this project, we would expect even better text generations that are able to be creative while being coherent and credible if we used a GPT-2 model with more parameters such as the large model that has 762M parameters or even more so if we had access to GPT-3. This can also be said about the BART model used for abstract summarization, given the resources and time, we would expect better results with the large model, and additionally, the library used for this task was a major drawback since it had many problems of memory allocation that didn't allow us to use our entire dataset.

Finally, in appendix B, all the individual results of each movie with respect to its title and description can be found. Moreover, in appendix A all the code used in the project.

6 Division of labor

In this group work, composed of 2 persons, each one partook in every task that was carried out in the project, but we could divide some tasks by the person that was responsible for each task:

- Writing of Team Abstract. Done by both teammates.
- Study of GPT-2 and its viability in the project. Done by both teammates.
- Study of BERT/BART and their viability in the project. Done by both teammates.
- Writing of Project Plan.
  - Architectures part done by Alicia.
  - Evaluation part done by Tudor.
  - Objective part done by Tudor and reviewed by Alicia.
- Data preprocessing.
  - Data loading by Tudor.
  - Filters done by Tudor and pipeline for filtering and saving data by Alicia.
- Implementation of GPT-2
  - Model implementation, loading and fine-tuning done by Alicia and Tudor.

  - Perplexity scores and generation parameters saved for each movie description done by Alicia.
- Implementation of BART.
  - Model implementation and loading done by Tudor.
  - Model fine-tuning done by Alicia and Tudor.
- Human Evaluation of GPT-2 and BART
  - Decision of metrics for human evaluation made by both.
- Movies form
  - Form created by Alicia and reviewed by Tudor.
- Writing of Project Report
  - Introduction written by Alicia and reviewed by Tudor.
  - Methods done by both teammates.
  - Experiments done by both teammates.
  - Results done by both teammates.
  - Conclusion done by both teammates.
  - Division of labor done by both teammates.
  - Appendices done by Alicia.

References

[1] Rafal Józefowicz et al. "Exploring the Limits of Language Modeling". In: *CoRR* abs/1602.02410 (2016). arXiv: 1602.02410. URL: http://arxiv.org/abs/1602.02410.

[2] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://www.aclweb.org/anthology/D14-1179.

[3] *Neural Machine Translation by Jointly Learning to Align and Translate*. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. 2014. URL: http://arxiv.org/abs/1409.0473.

[4] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[5] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *EMNLP*. 2020.

[6] Jay Alammar. *The Illustrated Transformer*. http://jalammar.github.io/illustrated-transformer/.

[7] Google AI Blog. *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing*. https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[8] A. Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[9] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[10] Sam Shleifer. *Introducing BART*. 2020. URL: https://sshleifer.github.io/blog_v2/jupyter/2020/03/12/bart.html.

[11] Ziang Xie. "Neural Text Generation: A Practical Guide". In: *CoRR* abs/1711.09534 (2017). arXiv: 1711.09534. URL: http://arxiv.org/abs/1711.09534.

[12] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. *Evaluation of Text Generation: A Survey*. 2020. arXiv: 2006.14799 [cs.CL].

[13] Anja Belz and Ehud Reiter. "Comparing Automatic and Human Evaluation of NLG Systems". In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy: Association for Computational Linguistics, Apr. 2006. URL: https://www.aclweb.org/anthology/E06-1040.

[14] Chris van der Lee et al. "Human evaluation of automatically generated text: Current trends and best practice guidelines". In: *Computer Speech Language* 67 (2021), p. 101151. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2020.101151. URL: https://www.sciencedirect.com/science/article/pii/S088523082030084X.

[15] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 2020. URL: https://web.stanford.edu/~jurafsky/slp3/.

[16] Kishore Papineni et al. *BLEU: a Method for Automatic Evaluation of Machine Translation*. 2002. URL: https://www.aclweb.org/anthology/P02-1040.pdf.

[17] Josef Steinberger and Karel Jezek. "Evaluation Measures for Text Summarization." In: *Computing and Informatics* 28 (Jan. 2009), pp. 251–275.

[18] Elena Lloret, Laura Plaza, and Ahmet Aker. *The Challenging Task of Summary Evaluation: An overview*. 2017. URL: https://core.ac.uk/download/pdf/132349065.pdf.

[19] Manik Bhandari et al. *Re-evaluating Evaluation in Text Summarization*. 2020. arXiv: 2010.07100 [cs.CL].

[20] Jun-Ping Ng and Viktoria Abrecht. "Better Summarization Evaluation with Word Embeddings for ROUGE". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1925–1930. DOI: 10.18653/v1/D15-1222. URL: https://www.aclweb.org/anthology/D15-1222.

[21] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[22] Sean Welleck et al. *Consistency of a Recurrent Language Model With Respect to Incomplete Decoding*. 2020. arXiv: 2002.02492 [cs.LG].

[23] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

[24] Pia Keukeleire. "Correspondence Between Perplexity Scores and Human Evaluation of Generated TV-Show Scripts". In: (). URL: http://resolver.tudelft.nl/uuid:ab543db3-f285-477c-b4ce-b6ac57507554.

Appendix A Code

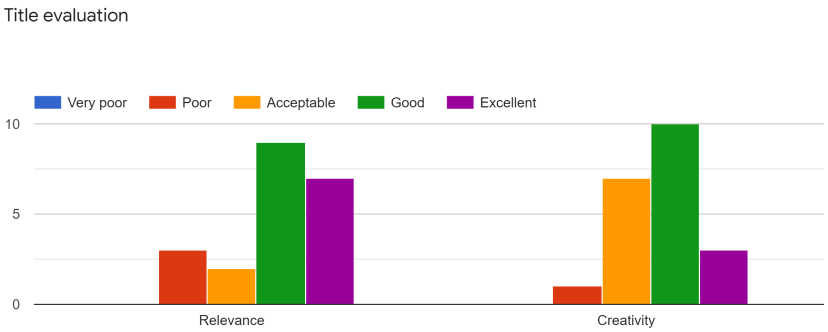
The project's code in in its entirety can be found in the following GitHub repository: https://github.com/Alicia6N/movie_generator

Moreover, the following Google Colab notebook is available where you can use the fine-tuned models to generate new descriptions and titles (it must be noted that the models can take a while to load): https://colab.research.google.com/drive/1M5ffWIso0FAnHlbqRvW0ow5aZAq091fn?usp=sharing.

Appendix B Individual movies survey results

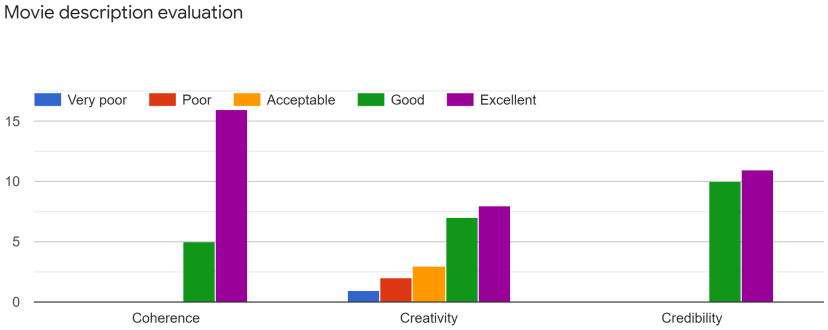
Title: Moonwalkers

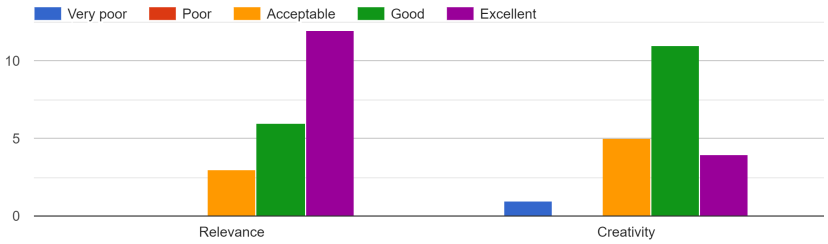
It is the year 2077 and space exploration has revived human interest in the moon. While mankind is busy building domed cities on the moon, a small mining colony is discovered on the other side of the galaxy.



Title: Dead Zone

After a zombie outbreak in Las Vegas, a group of survivors must cross the desert in an attempt to stay alive. Their only other contact is a mysterious radio tower that broadcasts a constant stream of cryptic messages.

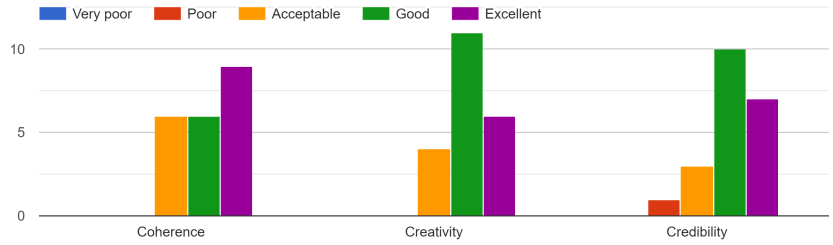




Title: Zentraedi

Humanity has mastered interplanetary spaceflight and begun to explore the galaxy. However, humanity's technological prowess has also brought new threats into the solar system, including a dangerous race of giant bugs. To combat these dangerous creatures, the government constructs the giant "Zentraedi" robots.

Movie description evaluation

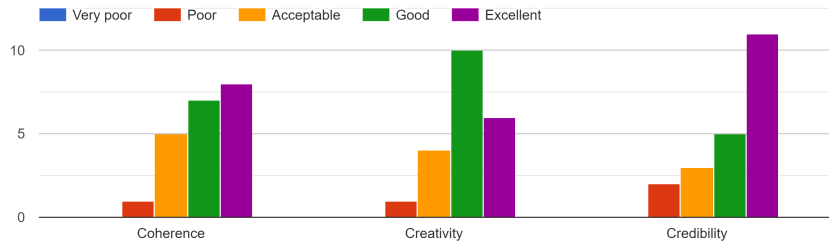


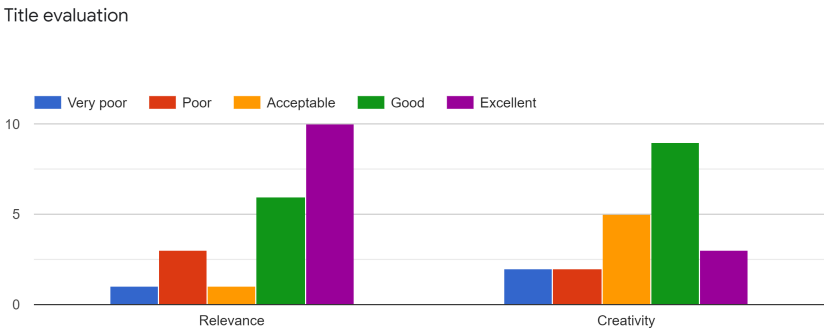
Title evaluation

Title: The Halsey Murders

After a series of harrowing murders in which a gang leader is savagely murdered and a local beauty is abducted, Inspector Jim Halsey finds evidence that points to a Mafia boss who has infiltrated his department - but he risks his own life by falling under the Mafia's tyrannical influence.

Movie description evaluation





Title: The Matrix

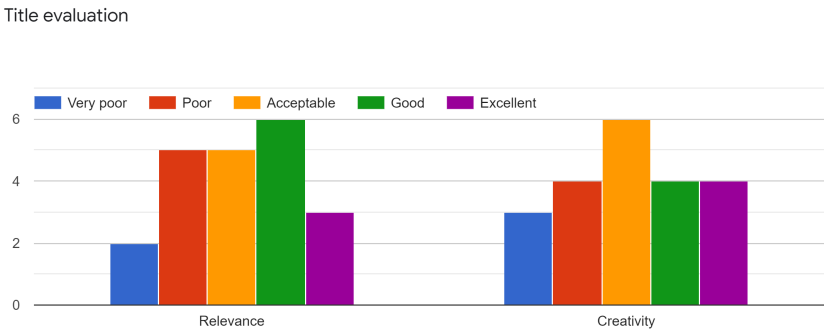
In the deep titanium mines on Alpha Centauri, scientists discover a fantastic alien invention: a mind matrix that can interface with any sentient device... including our bodies! But when the alien takes over the minds of local militia leaders, the military must investigate... and they find themselves battling a matrix that's out for blood!

Title: The Mandragora Syndrome

The human colony on Andromeda is attacked by a homicidal maniac and only Dr.Helmer, a doctor infected by the Mandragora virus, stands between their ship and destruction. In 1944, an elementary school teacher and his wife are crushed under the weight of a new marriage while on their road trip.

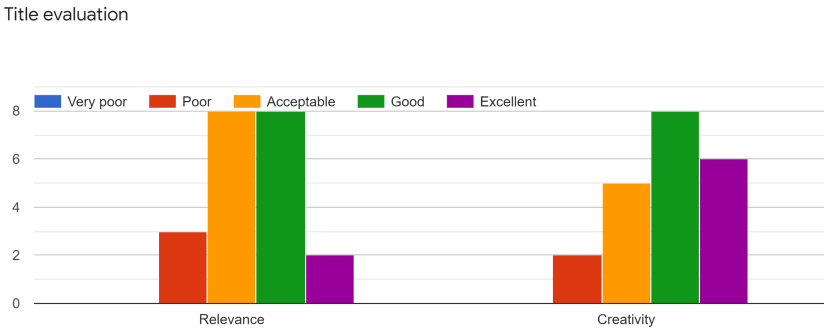
Title: The Mines of X

In the deep titanium mines on Alpha Centauri, two peace-seeking miners – Jim Randle and Bucky Hardin – attempt to work together in the face of relentless radiation, alien beasts, and an irradiated planet from which there is no escape. A troubled young man and woman move into and go to a rundown apartment house.



Title: Scaven

When five college kids arrive at a remote forest cabin to play a scavenger hunt, they soon find themselves in over their heads when they befriend a legendary, bloodsucking vampire who turns out to be more than they can handle.



Title: The Dark Star

The human colony on Andromeda is decimated by the ferocious, flesh-eating creatures that roam the galaxy seeking human flesh. To save the human race, the crew of the colony ship "Phoenix" must fight the creatures and their deadly weapon in this thrilling sci-fi action-adventure.

As a last question in the human evaluation form we wanted to know if any of the people that completed it had some background in NLP.